

Maximum Matching and Linear Programming in Fixed-Point Logic with Counting*

Matthew Anderson, Anuj Dawar, and Bjarki Holm

University of Cambridge Computer Laboratory

`firstname.lastname@cl.cam.ac.uk`

April 26, 2013

Abstract

We establish the expressibility in fixed-point logic with counting (FPC) of a number of natural polynomial-time problems. In particular, we show that the size of a maximum matching in a graph is definable in FPC. This settles an open problem first posed by Blass, Gurevich and Shelah [BGS99], who asked whether the existence of perfect matchings in general graphs could be determined in the more powerful formalism of choiceless polynomial time with counting. Our result is established by showing that the ellipsoid method for solving linear programs can be implemented in FPC. This allows us to prove that linear programs can be optimised in FPC if the corresponding separation oracle problem can be defined in FPC. On the way to defining a suitable separation oracle for the maximum matching problem, we provide FPC formulas defining maximum flows and canonical minimum cuts in capacitated graphs.

*Research supported by EPSRC grant EP/H026835. An extended abstract of this paper will appear in the proceedings of LICS 2013.

Contents

1	Introduction	3
2	Background	5
2.1	Logics and Structures	5
2.2	Numbers, Vectors and Matrices	7
2.3	Linear Programming	7
2.4	Representation	9
3	Expressing the Separation Problem in FPC	10
4	Reducing Optimisation to Separation in FPC	11
4.1	Folding	12
4.2	Folding Polytopes	15
4.3	Expressing Optimisation in FPC	16
5	Application: Maximum Flow	19
5.1	Expressing Maximum Flow in FPC	20
6	Application: Minimum Cut	20
6.1	Expressing Canonical Minimum Cut in FPC	21
7	Application: Minimum Odd Cut	22
7.1	Intersections of Minimum Cuts	23
7.2	Some Canonical Min (s, t) -Cuts are Min Odd Cuts	25
8	Application: Maximum Matching	29
8.1	Maximum Matching Program	29
8.2	Expressing Maximum Matching in FPC	30
9	Conclusion	31

1 Introduction

The question of whether there is a logical characterisation of the class P of problems solvable in polynomial time, first posed by Chandra and Harel [CH82], has been a central research question in descriptive complexity for three decades. At one time it was conjectured that FPC, the extension of inflationary fixed-point logic by counting terms, would suffice to express all polynomial-time properties, but this was refuted by Cai, Fürer and Immerman [CFI92]. Since then, a number of logics have been proposed whose expressive power is strictly greater than that of FPC but still contained within P . Among these are FPR, fixed-point logic with rank operators [DGHL09], and $\tilde{CPT}(\text{Card})$, choiceless polynomial time with counting [BGS99, BGS02]. For both of these it remains open whether their expressive power is strictly weaker than P .

Although it is known that FPC does not express all polynomial-time computable properties, the descriptive power of FPC still forms a natural class within P . For instance, it has been shown that FPC can express all polynomial-time properties on many natural graph classes, such as any class of proper minor-closed graphs [Gro10]. Delimiting the expressive power of FPC therefore remains an interesting challenge. In particular, it is of interest to establish what non-trivial polynomial-time algorithmic techniques can be expressed in this logic. The conjecture that FPC captures P was based on the intuition that the logic can define all “obvious” polynomial-time algorithms. The result of Cai et al. and the subsequent work of Atserias et al. [ABD09] showed that one important technique—that of Gaussian elimination for matrices over finite fields—is not captured by FPC. The question remains what other natural problems for which membership in P is established by non-trivial algorithmic methods might be expressible in FPC.

For instance, it was shown by Blass et al. [BGS02] that there is a sentence of FPC that is true in a bipartite graph G if, and only if, G contains a perfect matching. They posed as an open question whether the existence of a perfect matching on general graphs can be defined in $\tilde{CPT}(\text{Card})$ (see also [BG05, Ros10] for more on this open question). Indeed, this question first appears in [BGS99] where it is stated that it seems “unlikely” that this problem can be decided in $\tilde{CPT}(\text{Card})$. One of our main contributions in this paper is to settle this question by showing that the size of a maximum matching in a general graph can be defined in FPC (and therefore also in $\tilde{CPT}(\text{Card})$).

On the way to establishing this result, we show that a number of other interesting algorithmic problems can also be defined in FPC. To begin with, we study the logical definability of *linear programming* problems. Here we show that there is a formula of FPC which defines on a polytope a point inside the polytope which maximises a given linear objective function, if such a point exists. Here, by a polytope we mean a convex set in Euclidean space given by finite intersections of linear inequalities (or *constraints*) over a set of variables, suitably represented as a relational structure without an ordering on the sets of variables or constraints.

More specifically, we consider representations where, as in many applications of linear programming, the set of constraints is not given explicitly (indeed, it may be exponentially large) but is determined instead by a *separation oracle*. This is a procedure which, given a candidate point x , determines whether x is feasible and, if it is not, returns a constraint that is violated by x . It is well

known that Khachiyan’s polynomial-time algorithm for linear programming—the *ellipsoid method*—can be extended to prove that the linear programming and separation problems are polynomial-time equivalent (c.f., [Kha80, GLS81, GLS88]). We show an analogous result for FPC: if a separation oracle for a polytope is expressible in FPC, then linear programming on that polytope is definable in FPC. Informally speaking, the idea is the following. Although the set of variables is not inherently ordered, the separation oracle induces a natural equivalence relation on these variables whereby two variables are equivalent if they cannot be distinguished in any invocation of the oracle. Given a linear ordering on these equivalence classes, we can define in FPC a reduction of the optimisation problem to an instance with an ordered set of variables by taking the quotient of the polytope under the induced equivalence relation. We show that solving the optimisation problem on this quotiented polytope—now using the classical polynomial-time reduction made possible by the ordering—allows us to recover a solution to the original problem. In practice, neither the equivalence classes nor the order are given beforehand; rather they are iteratively refined via the invocations of the separation oracle made while optimising over the quotiented polytope. The details of this result are presented in Section 4.

Thus to express a problem modelled by a linear program in FPC it suffices to express a separation oracle in FPC. A key difficulty to expressing separation oracles in FPC is that a particular violated constraint must be chosen. We show in Section 3 that when the constraints are given explicitly, a *canonical* violated constraint can be defined by taking the sum of all the violated constraints. This implies that the class of feasible linear programs, when explicitly given, can be expressed in FPC. When the constraints are not given explicitly, it may still be possible to express canonical violated constraints (and hence separation oracles) by using domain knowledge.

As a first application of the FPC-definability of explicitly-given linear programs, we show that a maximum flow in a capacitated graph is definable in FPC. Indeed, this follows rather directly from the first result, since the flow polytope is of size polynomial in G and explicitly given, and hence a separation oracle can be easily defined from G in FPC. These results are presented in Section 5.

Next, we use the definability of maximum flows to show that minimum cuts are also definable in FPC. That is, in the vocabulary of capacitated graphs, there is a formula which defines a set of vertices C corresponding to a minimum value cut separating s from t . The cut C defined in this way is canonical in a strong sense, in that we show that it is the *smallest* (under set-inclusion) minimum cut separating s from t . The definition of minimum cut and useful variants are presented in Section 6.

Finally, we turn to the maximum matching problem. For a graph $G = (V, E)$, the matching polytope is given by a set of constraints of size exponential in the size of G . We show that there is a separation oracle for this set definable from G in FPC, using the definability of minimum cuts. To be precise, we use the fact that a separation oracle for the matching polytope can be obtained from a computation of minimum odd-size cuts in a graph [PR82]. In Section 7 we prove that there is always a pair of vertices s, t such that a canonical minimum (s, t) -cut is a minimum odd-size cut. This, combined with the definability of canonical minimum cuts, gives us the separation oracle for matching that we seek. Note that it is not possible in general to actually define a canonical

maximum matching. To see this, consider K_n , the complete graph on n vertices. This graph contains an exponential number of maximum matchings and for any two of these matchings, there is an automorphism of the graph taking one to the other. Thus, it is not possible for any formula of FPC (which is necessarily invariant under isomorphisms) to pick out a particular matching. What we can do, however, is to define a formula that gives the size of the maximum matching in a graph. This, in turn, enables us to write a sentence of FPC that is true in a graph G if, and only if, it contains a perfect matching. Our results on matchings are presented in Section 8.

2 Background

We write $[n]$ to denote the set of positive integers $\{0, \dots, n-1\}$. Given sets I and A , a column vector u over A indexed by I is a function $u : I \rightarrow A$, and we write A^I for the set of all such vectors. Similarly, an I, J -matrix over A is a function $M : I \times J \rightarrow A$ and we write M_{ij} for $M(i, j)$ and M_i for the row (vector) of M indexed by i . For an integer z , $|z|$ denotes its absolute value. For a vector $v \in \mathbb{Q}^I$, $\|v\|_\infty := \max_{i \in I} |v_i|$ denotes its infinity norm.

2.1 Logics and Structures

A relational *vocabulary* τ is a finite sequence of relation and constant symbols $(R_1, \dots, R_k, c_1, \dots, c_\ell)$, where every relation symbol R_i has a fixed *arity* $a_i \in \mathbb{N}$. A structure $\mathbf{A} = (\text{dom}(\mathbf{A}), R_1^\mathbf{A}, \dots, R_k^\mathbf{A}, c_1^\mathbf{A}, \dots, c_\ell^\mathbf{A})$ over the vocabulary τ (or a τ -*structure*) consists of a non-empty set $\text{dom}(\mathbf{A})$, called the *universe* of \mathbf{A} , together with relations $R_i^\mathbf{A} \subseteq \text{dom}(\mathbf{A})^{a_i}$ and constants $c_j^\mathbf{A} \in \text{dom}(\mathbf{A})$ for each $1 \leq i \leq k$ and $1 \leq j \leq \ell$. Members of the set $\text{dom}(\mathbf{A})$ are called the *elements* of \mathbf{A} and we define the *size* of \mathbf{A} to be the cardinality of its universe. In what follows, we often consider multi-sorted structures. That is, $\text{dom}(\mathbf{A})$ is given as the disjoint union of a number of different *sorts*. In this paper we consider only finite structures, that is structures over a finite universe. For a particular vocabulary τ we use $\text{fin}[\tau]$ to denote the set of all finite τ -structures.

Fixed-point logic with counting. Fixed-point logic with counting (FPC) is an extension of inflationary fixed-point logic with the ability to express the cardinality of definable sets. The logic has two types of first-order variable: *element variables*, which range over elements of the structure on which a formula is interpreted in the usual way, and *number variables*, which range over some initial segment of the natural numbers. We traditionally write element variables with lower-case Latin letters x, y, \dots and use lower-case Greek letters μ, η, \dots to denote number variables.

The atomic formulas of $\text{FPC}[\tau]$ are all formulas of the form: $\mu = \eta$ or $\mu \leq \eta$, where μ, η are number variables; $s = t$ where s, t are element variables or constant symbols from τ ; and $R(t_1, \dots, t_m)$, where each t_i is either an element variable or a constant symbol and R is a relation symbol of arity m . The set $\text{FPC}[\tau]$ of FPC *formulas* over τ is built up from the atomic formulas by applying an inflationary fixed-point operator $[\text{ifp}_{R, x} \phi](\vec{t})$; forming *counting terms* $\#_x \phi$, where ϕ is a formula and x an element variable; forming formulas of the kind $s = t$ and $s \leq t$ where s, t are number variables or counting terms; as well as the

standard first-order operations of negation, conjunction, disjunction, universal and existential quantification. Collectively, we refer to element variables and constant symbols as *element terms*, and to number variables and counting terms as *number terms*.

For the semantics, number terms take values in $[n + 1]$ and element terms take values in $\text{dom}(\mathbf{A})$ where $n := |\text{dom}(\mathbf{A})|$. The semantics of atomic formulas, fixed-points and first-order operations are defined as usual (c.f., e.g., [EF99] for details), with comparison of number terms $\mu \leq \eta$ interpreted by comparing the corresponding integers in $[n + 1]$. Finally, consider a counting term of the form $\#_x \phi$, where ϕ is a formula and x an element variable. Here the intended semantics is that $\#_x \phi$ denotes the number (i.e., the element of $[n + 1]$) of elements that satisfy the formula ϕ .

In general, a formula $\phi(\vec{x}, \vec{\mu})$ of FPC defines a relation over $\text{dom}(\mathbf{A}) \uplus [n + 1]$ that is invariant under automorphisms of \mathbf{A} . For a more detailed definition of FPC, we refer the reader to [EF99, Lib04].

It is known, by the results of Immerman and Vardi [Imm86, Var82], that every polynomial-time decidable property of *ordered* structures is definable in fixed-point logic, and therefore also in FPC. Here, an ordered structure is one which includes a binary relation which is a linear order of its universe. Throughout this paper, we refer to this result as the *Immerman-Vardi theorem*.

Logical interpretations. We frequently consider ways of defining one structure within another in some logic L , such as first-order logic or fixed-point logic with counting. Consider two vocabularies σ and τ and a logic L . An *m-ary L-interpretation of τ in σ* is a sequence of formulae of L in vocabulary σ consisting of: (i) a formula $\delta(\vec{x})$; (ii) a formula $\varepsilon(\vec{x}, \vec{y})$; (iii) for each relation symbol $R \in \tau$ of arity k , a formula $\phi_R(\vec{x}_1, \dots, \vec{x}_k)$; and (iv) for each constant symbol $c \in \tau$, a formula $\gamma_c(\vec{x})$, where each \vec{x} , \vec{y} or \vec{x}_i is an m -tuple of free variables. We call m the *width* of the interpretation. We say that an interpretation Θ associates a τ -structure \mathbf{B} to a σ -structure \mathbf{A} if there is a surjective map h from the m -tuples $\{\vec{a} \in (\text{dom}(\mathbf{A}) \uplus [n + 1])^m \mid \mathbf{A} \models \delta[\vec{a}]\}$ to \mathbf{B} such that:

- $h(\vec{a}_1) = h(\vec{a}_2)$ if, and only if, $\mathbf{A} \models \varepsilon[\vec{a}_1, \vec{a}_2]$;
- $R^{\mathbf{B}}(h(\vec{a}_1), \dots, h(\vec{a}_k))$ if, and only if, $\mathbf{A} \models \phi_R[\vec{a}_1, \dots, \vec{a}_k]$;
- $h(\vec{a}) = c^{\mathbf{B}}$ if, and only if, $\mathbf{A} \models \gamma_c[\vec{a}]$.

Note that an interpretation Θ associates a τ -structure with \mathbf{A} only if ε defines an equivalence relation on $(\text{dom}(\mathbf{A}) \uplus [n + 1])^m$ which is a congruence with respect to the relations defined by the formulae ϕ_R and γ_c . In such cases, however, \mathbf{B} is uniquely defined up to isomorphism and we write $\Theta(\mathbf{A}) := \mathbf{B}$.

It is not difficult to show that formulas of FPC compose with reductions in the sense that, given an interpretation Θ of σ in τ and a σ -formula ϕ , we can define a τ -formula ϕ' such that $\mathbf{A} \models \phi'$ if, and only if, $\Theta(\mathbf{A}) \models \phi$ (see [Imm99, Sec. 3.2]) In particular, if $\Theta(\mathbf{A})$ is an ordered structure, for all \mathbf{A} , then by the Immerman-Vardi theorem above, for any polynomial-time decidable class C , there is an FPC formula ϕ such that $\mathbf{A} \models \phi$ if, and only if, $\Theta(\mathbf{A}) \in C$.

2.2 Numbers, Vectors and Matrices

Let z be an integer, $b \geq \lceil \log_2(|z|) \rceil$, $B = [b]$ and write $\text{bit}(x, k)$ to denote the k -th least-significant bit in the binary expansion of $x \in \mathbb{N}$. We view the integer $z = s \cdot x$ as a product of a sign $s \in \{-1, 1\}$ and a natural number x . We can represent z as a single-sorted structure \mathbf{B} on a domain of bits B over the vocabulary $\tau_{\mathbb{Z}} := \{X, S, \leq_B\}$. Here \leq_B is interpreted as a linear ordering of B , the unary relation S indicates that the sign s of the integer is 1 if $S^{\mathbf{B}} = \emptyset$ and -1 otherwise, and the unary relation X is interpreted as $X^{\mathbf{B}} = \{k \in B \mid \text{bit}(x, k) = 1\}$. That is $k \in X^{\mathbf{B}}$ when the “the k -th bit in the binary expansion of x is 1.” Similarly we consider a *rational number* $q = s \cdot \frac{x}{d}$ as a structure on the domain of bits B over $\tau_{\mathbb{Q}} := \{X, D, S, \leq_B\}$, where X and S are as before and D is interpreted as the binary encoding of the denominator d when $D^{\mathbf{B}} \neq \emptyset$.

We now generalise these notions and consider unordered tensors over the rationals (the case of integers is completely analogous). Let J_1, \dots, J_r be a family of finite non-empty sets. An unordered tensor T over \mathbb{Q} is a function $T : J_1 \times \dots \times J_r \rightarrow \mathbb{Q}$. We write $t_{j_1 \dots j_r} = s_{j_1 \dots j_r} \frac{x_{j_1 \dots j_r}}{d_{j_1 \dots j_r}}$ to denote the element of T indexed by $(j_1, \dots, j_r) \in J_1 \times \dots \times J_r$. Writing $m \in \mathbb{N}$ for the the maximum absolute value of integers appearing as either numerators or denominators of elements in the range of T , let $b \geq \lceil \log_2(|m|) \rceil$ and $B = [b]$. The tensor T is then an $(r+1)$ -sorted structure \mathbf{T} with r index sorts J_1, \dots, J_r and a bit sort B over the vocabulary $\sigma_{\text{ten}, r} := \{X, D, S, \leq_B\}$. Here \leq_B is interpreted as before, the $(r+1)$ -ary relation S is interpreted as indicating the value of the sign $s_{j_1 \dots j_r} \in \{-1, 1\}$ as before, the $(r+1)$ -ary relation X is interpreted as

$$\{(j_1, \dots, j_r, k) \in J_1 \times \dots \times J_r \times B \mid \text{bit}(x_{j_1 \dots j_r}, k) = 1\},$$

and the $(r+1)$ -ary relation D is similarly interpreted as the binary representation of the denominators of T . We are only interested in the case of rational vectors and matrices and so define the vocabularies $\tau_{\text{vec}} := \sigma_{\text{ten}, 1}$ and $\tau_{\text{mat}} := \sigma_{\text{ten}, 2}$.

In [Hol10] it is shown that a variety of basic linear-algebraic operations on rational vectors and matrices described in this way can be expressed in fixed-point logic with counting. These include computing equality, norms, dot product, matrix product, determinant and inverse.

2.3 Linear Programming

We recall some basic definitions from combinatorics and linear optimisation. For further background, see, for example, the textbook by Grötschel et al. [GLS88].

Polytopes. Consider the rational Euclidean space \mathbb{Q}^V indexed by a set V . The solutions to a system of linear equalities and inequalities over \mathbb{Q}^V is the intersection of some number of *half-spaces* of the kind $\{x \in \mathbb{Q}^V \mid a^\top x \leq b\}$ specified by the *constraint* $a^\top x \leq b$, where $a \in \mathbb{Q}^V$ and $b \in \mathbb{Q}$. A (rational) *polytope* is a convex set $P \subseteq \mathbb{Q}^V$ which is the intersection of a *finite* number of half-spaces. That is to say, there are a set of constraints C , a *constraint matrix* $A \in \mathbb{Q}^{C \times V}$ and vector $b \in \mathbb{Q}^C$, such that $P = P_{A, b} := \{x \in \mathbb{Q}^V \mid Ax \leq b\}$.

Polytopes have an alternative characterisation as a combination of convex hulls and cones. Let S be a finite set of points in \mathbb{Q}^V and define the *convex hull*

of S

$$\text{conv}(S) := \left\{ \sum_{s \in S} \lambda_s s \mid \lambda_s \in \mathbb{Q}_{\geq 0}, \forall s \in S \text{ and } \sum_{s \in S} \lambda_s = 1 \right\},$$

and similarly define the *cone* of S

$$\text{cone}(S) := \left\{ \sum_{s \in S} \lambda_s s \mid \lambda_s \in \mathbb{Q}_{\geq 0}, \forall s \in S \right\}.$$

If P is a polytope in \mathbb{Q}^V , then there exist finite sets $S_1, S_2 \subseteq \mathbb{Q}^V$ such that $P = P_{S_1, S_2} := \text{conv}(S_1) + \text{cone}(S_2) = \{x_1 + x_2 \mid x_1 \in \text{conv}(S_1), x_2 \in \text{cone}(S_2)\}$.

The *size*, or bit complexity, of a vector $c \in \mathbb{Q}^V$ (denoted $\langle c \rangle$) is the number of bits required to encode the components of c in some standard encoding of rational numbers. Note that $\langle c \rangle$ is at least $|V|$. The size of a constraint $a^\top x \leq b$ is then $\langle \begin{pmatrix} a \\ b \end{pmatrix} \rangle$. If $Ax \leq b$ is a system of linear inequalities then its size is the maximum over the sizes of its individual constraints. Note that this measure is explicitly independent of the number of constraints in the system. The *facet complexity* $\langle P \rangle_f$ of a polytope P is the minimum over the sizes of the systems $Ax \leq b$ such that $P = P_{A, b}$. The *vertex complexity* $\langle P \rangle_v$ of a polytope P is the minimum over the maximum size vector in the union of sets $S_1, S_2 \subseteq \mathbb{Q}^V$ such that $P = \text{conv}(S_1) + \text{cone}(S_2)$. The facet and vertex complexity of a polytope are closely related: $\langle P \rangle_v \leq 4|V|^2 \langle P \rangle_f$ and $\langle P \rangle_f \leq 3|V|^2 \langle P \rangle_v$ [GLS88, Lemma 6.2.4].

Problems on polytopes. We are interested in two main combinatorial problems on polytopes: *linear optimisation* and *separation*.

Problem 1 (Linear Optimisation). *Let V be a set, $P \subseteq \mathbb{Q}^V$ be a polytope and $c \in \mathbb{Q}^V$. The linear optimisation problem on P is the problem of determining either (i) an element $y \in P$ such that $c^\top y = \max\{c^\top x \mid x \in P\}$, (ii) that $P = \emptyset$ or (iii) that P is unbounded in the direction of c .*

An instance of the linear optimisation problem is called a *linear program* and the linear function $x \mapsto c^\top x$ is called the *objective function*. Over the years, a number of algorithms for solving linear programs have been studied. Early work by Dantzig [Dan63] gave a combinatorial algorithm—the *simplex method*—which traverses the vertices (extremal points) of the polytope favouring vertices that improve the objective value. Although the simplex method is useful in practice, it tends not to be theoretically useful because strong worst-case performance guarantees are not known¹. A series of works studying linear programming from a geometric perspective [Sho72, YN76, Sho77] culminated in the breakthrough of Khachiyan [Kha79, Kha80] which established a polynomial-time algorithm—the *ellipsoid method*—for solving linear programs. One of the strengths of the ellipsoid method is that it can be applied to linear programs where the constraints are not given explicitly. In such implicitly-defined linear programs, we are instead given a polynomial-time algorithm, known as a *separation oracle*, for solving the following “separation problem”.

¹Stronger guarantees are known for the average-case and smoothed complexity of the simplex method [ST04].

Problem 2 (Separation). *Let V be a set, $P \subseteq \mathbb{Q}^V$ be a polytope and $y \in \mathbb{Q}^V$. The separation problem on P is the problem of determining either (i) that $y \in P$ or (ii) a vector $c \in \mathbb{Q}^V$ with $c^\top y > \max\{c^\top x \mid x \in P\}$ and $\|c\|_\infty = 1$.*

Over families of rational polytopes, the optimisation and separation problems are polynomial-time equivalent (c.f., e.g., [GLS88, Theorem 6.4.9]). Here the time bound is measured in the size of the polytope and all other parameters of the problem.

2.4 Representation

When we deal with polytopes as objects in a computation, we need to choose a representation which gives a finite description of a polytope. In particular, in dealing with logical definability of problems on polytopes, we need to choose a representation of polytopes by relational structures.

Definition 3. *A representation of a class \mathcal{P} of polytopes is a relational vocabulary τ along with an onto function $\nu : \text{fin}[\tau] \rightarrow \mathcal{P}$ which is isomorphism invariant, that is, $\mathbf{A} \cong \mathbf{B}$ implies $\nu(\mathbf{A}) \cong \nu(\mathbf{B})$.*

For concreteness, consider the vocabulary $\tau := \tau_{\text{mat}} \uplus \tau_{\text{vec}}$ obtained by taking the disjoint union of the vocabularies for rational matrices and vectors. A τ -structure over a universe consisting of a set V of variables and a set C of constraints describes a constraint matrix $A \in \mathbb{Q}^{C \times V}$ and bound vector $b \in \mathbb{Q}^C$. Thus, the function taking such a structure to the polytope $P_{A,b}$ is a representation of the class of rational polytopes. We call this the *explicit representation*.

Note that the explicit representation of polytopes has the property that both the size of the polytope (i.e., the maximum size of any constraint) and the number of constraints of $\nu(\mathbf{A})$ are polynomially bounded in the size of \mathbf{A} . We will also be interested in representations ν where the number of constraints in $\nu(\mathbf{A})$ is exponential in $|\mathbf{A}|$, but we always confine ourselves to representations where the size of the constraints is bounded by a polynomial in \mathbf{A} . We formalise this by saying that a representation ν is *well described* if there is a polynomial p such that $\langle \nu(\mathbf{A}) \rangle = p(|\mathbf{A}|)$, for all τ -structures \mathbf{A} . In particular, in all representations we consider the *dimension* of the polytope $\nu(\mathbf{A})$ is bounded by a polynomial in $|\mathbf{A}|$.

We are now ready to define what it means to express the linear optimisation and separation problems in FPC.

Definition 4. *We say that the linear optimisation problem for a class of polytopes \mathcal{P} is expressible in FPC with respect to a representation $\nu : \text{fin}[\tau] \rightarrow \mathcal{P}$ if there is an FPC interpretation of $\tau_{\mathbb{Q}} \uplus \tau_{\text{vec}}$ in $\tau \uplus \tau_{\text{vec}}$ which takes a τ -structure \mathbf{A} and a vector c to a rational f and vector y such that either (i) $f = 1$, and $\nu(\mathbf{A})$ is unbounded in the direction of c , or (ii) $f = 0$, and $\nu(\mathbf{A}) \neq \emptyset$ iff $y \in \nu(\mathbf{A})$ and $c^\top y = \max\{c^\top x \mid x \in \nu(\mathbf{A})\}$.*

Definition 5. *The separation problem for a class of polytopes \mathcal{P} is expressible in FPC with respect to a representation $\nu : \text{fin}[\tau] \rightarrow \mathcal{P}$ if there is an FPC interpretation of τ_{vec} in $\tau \uplus \tau_{\text{vec}}$ which takes a structure coding a τ -structure \mathbf{A} and a vector y to a vector c such that either (i) $y \in \nu(\mathbf{A})$ and $c = 0$, or (ii) $c \in \mathbb{Q}^V$ with $c^\top y > \max\{c^\top x \mid x \in \nu(\mathbf{A})\}$ and $\|c\|_\infty = 1$.*

Figure 1: A separation oracle for explicitly-represented rational polytopes.

$$\Delta(A, b, x)$$

Input: $A \in \mathbb{Q}^{C \times V}$, $b \in \mathbb{Q}^C$ and $x \in \mathbb{Q}^V$.
Output: $c \in \mathbb{Q}^V$ solving the separation problem for the polytope $P_{A,b}$ and x .

- 1: **if** $Ax \leq b$ **then return** 0^V .
- 2: Select $k \in C$ such that $A_k x > b_k$.
- 3: **return** $\frac{A_k}{\|A_k\|_\infty}$.

3 Expressing the Separation Problem in FPC

Let $A \in \mathbb{Q}^{C \times V}$ be a constraint matrix and $b \in \mathbb{Q}^C$ a constraint vector of the polytope $P_{A,b}$. Figure 1 presents a straightforward algorithm (Δ) for solving the separation problem for the explicitly-represented polytope $P_{A,b}$. It is not hard to see that the algorithm Δ can be implemented in time polynomial in the size of the explicit natural representation of inputs A, b and x .

If we try to express the algorithm Δ in fixed-point logic with counting, we first note that we can define in FPC all the relevant manipulations on rational values, vectors and matrices, such as norms, addition and multiplication [Hol10], even when they are indexed by unordered sets. This shows that both lines 1 and 3 of the algorithm can be simulated in FPC. However, line 2 poses a problem as the logic is in general not able to *choose* a particular element from an unordered set. Our key observation here is that linearity implies that the sum of all such violated constraints is itself a violated constraint for non-empty polytopes and hence the choice made by Δ is superfluous. This can be formally stated as follows.

Proposition 6. *Let $A \in \mathbb{Q}^{C \times V}$, $b \in \mathbb{Q}^C$, $x \in \mathbb{Q}^V$ and $C \supseteq S \neq \emptyset$. Suppose $P_{A,b}$ is non-empty and $(Ax)_s \not\leq b_s$ for all $s \in S$. Define $a_S := \sum_{s \in S} A_s$. Then $a_S^\top x > \max\{a_S^\top y \mid y \in P_{A,b}\}$ and $a_S \neq 0^V$.*

Proof. Define $b_S := \sum_{s \in S} b_s$. That $a_S^\top x > b_S$ is immediate from linearity. Since the polytope is non-empty pick any point $y \in P_{A,b}$. By definition, $Ay \leq b$. Linearity implies that $a_S^\top y \leq b_S$. Thus $a_S^\top x > b_S \geq \max\{a_S^\top y \mid y \in P_{A,b}\}$. This also implies that $a_S \neq 0^V$. \square

This observation leads to a definition in FPC of the separation problem for $P_{A,b}$ with respect to x . Specifically, let $S \subseteq C$ be the set of constraints which violate the inequality $Ax \leq b$. This set can be defined by a FPC formula using rational arithmetic. If S is empty, expressing $c = 0^V$ correctly indicates that $x \in P_{A,b}$. Otherwise S is non-empty; let a_S be the sum of the constraints which x violates. Since the set S is definable in FPC so is the sum of constraints indexed by S . If $a_S \neq 0^V$, Proposition 6 implies that expressing c as the division of a_S by its (non-zero) infinity norm correctly indicates a separating hyperplane for $P_{A,b}$ through x ; moreover, both operations are in FPC. Otherwise, $a_S = 0^V$ and Proposition 6 indicates that $P_{A,b}$ is empty. This means that any non-zero vector defines a separating hyperplane for $P_{A,b}$. Thus it suffices for the interpretation to express the vector $c = 1^V$. Overall, the above discussion gives us a proof of the following theorem.

Theorem 7. *There is an FPC interpretation of τ_{vec} in $\tau_{mat} \uplus \tau_{vec} \uplus \tau_{vec}$ expressing the separation problem for the class of polytopes explicitly given by constraints and represented naturally as $\tau_{mat} \uplus \tau_{vec}$ -structures.*

4 Reducing Optimisation to Separation in FPC

In this section we present our main technical result, which is an FPC reduction from optimisation to separation, which treats the classical polynomial-time reduction of the corresponding problems as a subroutine. This classical result can be stated as follows.

Theorem 8 (c.f., e.g., [GLS88, Theorem 6.4.9]²). *The linear optimisation problem can be solved in polynomial time for any well-described polytope given by a polynomial-time oracle solving the separation problem for that polytope.*

Below, we prove the following analogous result for fixed-point logic with counting.

Theorem 9 (Optimisation to Separation). *Let \mathcal{P} be a class of well-described rational polytopes represented by τ -structures and the function ν . Let Σ be an FPC interpretation of τ_{vec} in $\tau \uplus \tau_{vec}$ expressing the separation problem for \mathcal{P} with respect to ν . Then there is an FPC interpretation of $\tau_{\mathbb{Q}} \uplus \tau_{vec}$ in $\tau \uplus \tau_{vec}$ which expresses the linear optimisation problem for \mathcal{P} with respect to ν .*

Observe that these theorems do not imply that every linear optimisation problem can be solved in FPC (or even in polynomial time). Rather one can solve particular classes of linear optimisation problems where domain knowledge can be used to solve the separation problem. We have the following generic consequence in the case of explicitly-given polytopes when Theorem 9 is combined with Theorem 7.

Theorem 10 (Explicit Optimisation). *There is an FPC-interpretation of $\tau_{\mathbb{Q}} \uplus \tau_{vec}$ in $\tau_{mat} \uplus \tau_{vec} \uplus \tau_{vec}$ expressing the linear optimisation problem for the class of polytopes explicitly given by constraints and represented naturally as $\tau_{mat} \uplus \tau_{vec}$ -structures.*

The main idea behind the proof of Theorem 9 is as follows. Suppose we are given a polytope $P \subseteq \mathbb{Q}^V$ by an FPC-interpretation Σ_P that expresses the separation problem for P . A priori the elements of V are indistinguishable. However, Σ_P may expose an underlying order in V as it expresses answers to the separation problem for P . For example, suppose Σ_P on some input expresses a vector $d \in \mathbb{Q}^V$ where the components d_u and d_v for $u, v \in V$ are different. This information can be used to distinguish the components u and v ; moreover, it can be used to order the components because d_u and d_v are distinct elements of a field with a total order. As Σ_P is repeatedly used it may expose more and more information about the asymmetry of P . This partial information can be represented by maintaining a sequence of equivalence classes $(V_i)_{i=1}^k$ partitioning V . This equivalence relation is progressively refined through further invocations of the separation oracle. Initially all elements of V reside in a single class.

²The reverse of this theorem also holds: An oracle for the linear optimisation problem can be used to solve the separation problem.

It is natural to consider the polytope P' derived from P by taking its quotient under the equivalence relation defined in this way. Intuitively, this maps polytopes in \mathbb{Q}^V to polytopes in \mathbb{Q}^k by summing the components in each equivalence class to form a single new component which is ordered by the sequence. We call this process *folding*. We observe that a separation oracle for P' can be constructed using Σ_P , provided the answers of Σ_P never expose more asymmetry than was used to derive P' . However, failing to meet this proviso is informative—it further distinguishes the elements of V —and refines the sequence of equivalence classes.

These observations suggest the following algorithm. Start with a sequence (V_1) of exactly one class which contains all of V . Construct the folded polytope P' with respect to this sequence and the associated separation oracle $\Sigma_{P'}$ from Σ_P . Attempt to solve the linear optimisation problem on the folded polytope (which lies in an ordered space) using the Immerman-Vardi theorem [Var82, Imm86] and the classical polynomial-time reduction from optimisation to separation (Theorem 8)³. Should Σ_P at any point answer with a vector that distinguishes more elements of V than the current sequence of equivalence classes, then we: (i) abort the run; (ii) refine the equivalence classes and the folded polytope with this new information; and, finally, (iii) restart the optimisation procedure on this more representative problem instance. Since the number of equivalence classes increases each time the algorithm aborts, it eventually solves the optimisation problem for some P' without aborting. We argue that this solution for P' can be translated into a solution for P .

A key aspect of this approach is that it treats the polynomial-time reduction from optimisation to separation as a blackbox, i.e., it assumes nothing about how the reduction works internally.⁴ Before formally describing the algorithm we establish a number of useful definitions and technical properties.

4.1 Folding

Let V be a set. For $k \leq |V|$, let $\sigma : V \rightarrow [k]$ be an onto map. We call σ an *index map*. For $i \in [k]$ define $V_i := \{s \in V \mid \sigma(s) = i\}$. The sequence of sets V_i is a partition of V .

Definition 11 (Folding).

For a vector $x \in \mathbb{Q}^V$ let the almost-folded vector $[x]^{\tilde{\sigma}}$ of \mathbb{Q}^k be given by

$$([x]^{\tilde{\sigma}})_i := \sum_{v \in V_i} x_v, \text{ for } i \in [k].$$

For a vector $x \in \mathbb{Q}^V$ let the folded vector $[x]^\sigma$ of \mathbb{Q}^k be given by

$$([x]^\sigma)_i := \frac{[x]^{\tilde{\sigma}}_i}{|V_i|}, \text{ for } i \in [k].$$

For a vector $x \in \mathbb{Q}^k$ let the unfolded vector $[x]^{-\sigma}$ of \mathbb{Q}^V be given by

$$([x]^{-\sigma})_v := x_i, \text{ with } V_i \ni v, \text{ for } v \in V.$$

³Recall that, by the Immerman-Vardi theorem, every polynomial-time property of ordered structures is definable in fixed-point logic, and hence also in FPC.

⁴It is, in fact, possible to translate the classical reduction from optimisation to separation line by line into FPC (in the spirit of Section 3). However, this translation quickly becomes mired in intricate error analysis which is both tedious and opaque.

We say a vector $x \in \mathbb{Q}^V$ *agrees with* σ when for all $v, v' \in V$, $\sigma(v) = \sigma(v')$ implies $x_v = x_{v'}$. It easily follows that if x agrees with σ then $[[x]^\sigma]^{-\sigma} = x$. When vectors agree with σ and σ is clear from context we often use the font, as above with x and \mathbf{x} , to indicate whether a vector is unfolded and lies in \mathbb{Q}^V , or folded and lies in \mathbb{Q}^k , respectively. The notion of folding naturally extends to a set $S \subseteq \mathbb{Q}^V$ (and hence polytopes): Let $[S]^\sigma := \{[s]^\sigma \mid s \in S\}$. See Figures 2 and 3 for examples of folding polytopes. Note that $[P]^\sigma$ is a projection of P into the k -dimensional space \mathbb{Q}^k .

Several useful properties of folding and unfolding follow directly from their definitions.

Proposition 12. *Let $\sigma : V \rightarrow [k]$ be an index map and $c, x \in \mathbb{Q}^V$ such that c agrees with σ . Then,*

$$c^\top [[x]^\sigma]^{-\sigma} = c^\top x = [c]^{\tilde{\sigma}^\top} [x]^\sigma.$$

Proof. We begin by proving the first equality. Fix $i \in [k]$. Definition 11 implies that

$$\begin{aligned} \sum_{v \in V_i} ([x]^\sigma)^{-\sigma}_v - x_v &= \sum_{v \in V_i} ([x]^\sigma)_i - \sum_{v \in V_i} x_v = \sum_{v \in V_i} \left(\frac{1}{|V_i|} \sum_{v' \in V_i} x_{v'} \right) - \sum_{v \in V_i} x_v \\ &= \frac{|V_i|}{|V_i|} \sum_{v' \in V_i} x_{v'} - \sum_{v \in V_i} x_v = 0. \end{aligned} \tag{1}$$

We conclude that

$$\begin{aligned} c^\top ([x]^\sigma)^{-\sigma} - x &= \sum_{v \in V} c_v (([x]^\sigma)^{-\sigma})_v - x_v \\ &= \sum_{i \in [k]} \sum_{v \in V_i} c_v (([x]^\sigma)^{-\sigma})_v - x_v && V = \uplus_{i \in [k]} V_i \\ &= \sum_{i \in [k]} \sum_{v \in V_i} \frac{([c]^{\tilde{\sigma}})_i}{|V_i|} (([x]^\sigma)^{-\sigma})_v - x_v && c \text{ agrees with } \sigma \\ &= \sum_{i \in [k]} \frac{([c]^{\tilde{\sigma}})_i}{|V_i|} \sum_{v \in V_i} (([x]^\sigma)^{-\sigma})_v - x_v && \text{linearity} \\ &= \sum_{i \in [k]} \frac{([c]^{\tilde{\sigma}})_i}{|V_i|} \cdot 0 = 0. \end{aligned} \tag{1}$$

We now argue the second equality.

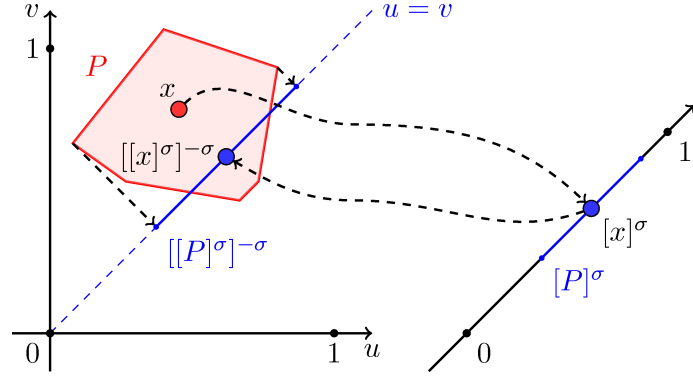


Figure 2: Folding and unfolding a polytope $P \subseteq \mathbb{Q}^{\{u,v\}}$ with respect to $\sigma = \{u \rightarrow 0, v \rightarrow 0\}$.

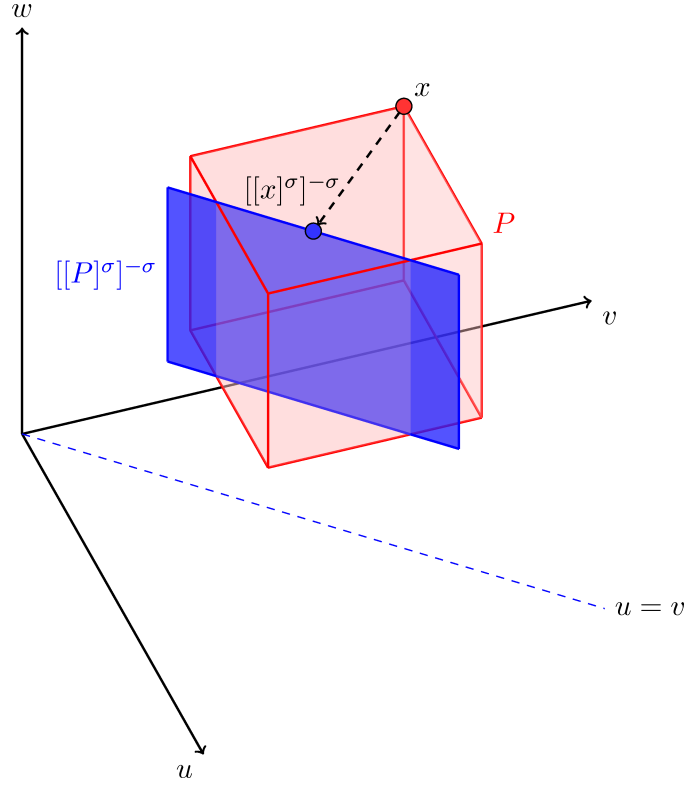


Figure 3: Folding and unfolding a polytope $P \subseteq \mathbb{Q}^{\{u,v,w\}}$ with respect to $\sigma = \{u \rightarrow 0, v \rightarrow 0, w \rightarrow 1\}$.

$$\begin{aligned}
c^\top x &= \sum_{v \in V} c_v x_v \\
&= \sum_{i \in [k]} \sum_{v \in V_i} c_v x_v && V = \uplus_{i \in [k]} V_i \\
&= \sum_{i \in [k]} \sum_{v \in V_i} \left(\frac{1}{|V_i|} \sum_{v' \in V_i} c_{v'} \right) x_v && c \text{ agrees with } \sigma \\
&= \sum_{i \in [k]} \left(\sum_{v' \in V_i} c_{v'} \right) \left(\frac{1}{|V_i|} \sum_{v \in V_i} x_v \right) && \text{linearity} \\
&= \sum_{i \in [k]} ([c]^{\tilde{\sigma}})_i ([x]^\sigma)_i && \text{Def. 11} \\
&= [c]^{\tilde{\sigma}^\top} [x]^\sigma.
\end{aligned}$$

□

4.2 Folding Polytopes

The diagrams in Figures 2 and 3 suggest intuitively that the result of folding a polytope is itself a polytope; the following proposition makes this connection concrete.

Proposition 13. *Let P be a polytope in \mathbb{Q}^V and let $\sigma : V \rightarrow [k]$ be an index map. Then the folded set $[P]^\sigma$ is a polytope with $\langle [P]^\sigma \rangle_f \leq 48k^3 |V|^3 \langle P \rangle_f$.*

Proof. Let $P = \text{conv}(S_1) + \text{cone}(S_2)$ for two finite sets of points $S_1, S_2 \subseteq \mathbb{Q}^V$. By the linearity of $[\cdot]^\sigma$ we have

$$\begin{aligned}
[P]^\sigma &= [\text{conv}(S_1) + \text{cone}(S_2)]^\sigma \\
&= [\text{conv}(S_1)]^\sigma + [\text{cone}(S_2)]^\sigma \\
&= \text{conv}([S_1]^\sigma) + \text{cone}([S_2]^\sigma).
\end{aligned}$$

We conclude that $[P]^\sigma$ is a polytope. We have

$$\langle [P]^\sigma \rangle_f \leq 3k^2 \langle [P]^\sigma \rangle_v \leq 3k^2 \cdot 4k |V| \langle P \rangle_v \leq 12k^3 |V| \cdot 4|V|^2 \langle P \rangle_f$$

where the middle inequality comes from bounding the bit complexity of $\frac{1}{|V|}v$ for extremal vertices $v \in P$. □

For a polytope $P \subseteq \mathbb{Q}^V$ and a point $x \in \mathbb{Q}^V$ (with $x \notin P$) we say that *all separating hyperplanes at x disagree with σ* if there is no $c \in \mathbb{Q}^V$ which both agrees with σ and has $c^\top x > \max\{c^\top y \mid y \in P\}$. This induces an alternative characterisation of the polytope $[P]^\sigma$.

Lemma 14. *Let P be a polytope in \mathbb{Q}^V and $\sigma : V \rightarrow [k]$ be an index map. Then*

$$[P]^\sigma = P' := \left\{ x \in \mathbb{Q}^k \mid \begin{array}{l} [x]^{-\sigma} \in P \text{ or all separating hyperplanes} \\ \text{at } [x]^{-\sigma} \text{ disagree with } \sigma \end{array} \right\}.$$

Proof. We show both inclusions.

1. $[P]^\sigma \subseteq P'$: Let $x \in [P]^\sigma$. By definition there is a point $x \in P$ such that $[x]^\sigma = x$. Suppose $x = [x]^{-\sigma}$, then $[x]^{-\sigma} \in P$ and hence $x \in P'$. Thus assume $[x]^{-\sigma} \neq x$. Let $c \in \mathbb{Q}^V$ be any vector agreeing with σ . By Proposition 12 we have $c^\top [x]^{-\sigma} = c^\top [[x]^\sigma]^{-\sigma} = c^\top x$. Since $x \in P$, c is not the normal of a separating hyperplane through $[x]^{-\sigma}$. We conclude that all separating hyperplanes through $[x]^{-\sigma}$ disagree with σ and hence that $x \in P'$.

2. $[P]^\sigma \supseteq P'$: Let $x \in P'$. Suppose $[x]^{-\sigma} \in P$, then $x = [[x]^{-\sigma}]^\sigma \in [P]^\sigma$. Thus assume that $[x]^{-\sigma} \notin P$ and that all separating hyperplanes through $[x]^{-\sigma}$ disagree with σ . This means that for any vector $c \in \mathbb{Q}^V$ that agrees with σ the hyperplane through $[x]^{-\sigma}$ with normal c intersects P and thus there is a point $y \in P$ which has $c^\top [x]^{-\sigma} = c^\top y$. This further implies that $c^\top [x]^{-\sigma} \leq \max\{c^\top y \mid y \in P\}$. Since c agrees with σ , Proposition 12 implies that

$$[c]^{\tilde{\sigma}}^\top x \leq \max\{[c]^{\tilde{\sigma}}^\top [y]^\sigma \mid y \in P\}.$$

Observe $\{c \in \mathbb{Q}^V \mid c \text{ agrees with } \sigma\}^{\tilde{\sigma}} = \mathbb{Q}^k$. This means for any vector $c' \in \mathbb{Q}^k$, $c'^\top x \leq \max\{c'^\top [y]^\sigma \mid y \in P\}$. In particular, for every constraint defining the polytope $[P]^\sigma$, x also satisfies that constraint. We conclude that $x \in [P]^\sigma$. \square

4.3 Expressing Optimisation in FPC

Suppose we are given a polytope $P \subseteq \mathbb{Q}^V$ via a separation oracle Δ_P , and a vector c indicating a linear objective. The algorithm maintains an index map $\sigma : V \rightarrow [k]$ that indicates a sequence of equivalence classes of V which have not been distinguished by the algorithm so far. Initially this index map is given by ordering variables according to their relative values in c . Under the assumption that σ accurately describes the symmetries of P we execute the polynomial-time reduction from optimisation to separation on the polytope $[P]^\sigma$ and objective $[c]^{\tilde{\sigma}}$. Since $[P]^\sigma$ lies in an ordered space, it follows from the Immerman-Vardi theorem that the reduction can be expressed in fixed-point logic with counting.

To this end, a separation oracle $\Delta_{[P]^\sigma}$ must be specified for the polytope $[P]^\sigma$. Given a point $x \in \mathbb{Q}^k$, we argue that the result of applying Δ_P to the unfolding of x either determines the point is in P , and hence also in $[P]^\sigma$; or determines a separating hyperplane for P . If a separating hyperplane is determined, it can be folded into a separating hyperplane for $[P]^\sigma$, but only if the hyperplane normal agrees with σ . In the case the separating hyperplane disagrees with σ , our assumption about P is violated, and our separation oracle does not have enough information to proceed. Indeed, folding the resulting normal may produce 0^k which is not a valid answer. In this case, the algorithm aborts the run of the linear optimisation algorithm, and returns the disagreeing hyperplane normal. The algorithm then combines the disagreeing normal with its current index map σ to produce a new index map which is consistent with σ and agreed with by the disagreeable hyperplane normal. This strictly increases the number of equivalence classes of variables induced by the index map. The above procedure can abort at most $|V|$ times before σ exactly characterises the order of V relative to P . After this point the linear optimisation algorithm cannot abort and hence must solve the optimisation problem for $[P]^\sigma$ which can be unfolded into a solution for P .

Figure 4: An instrumentation of the reduction from optimisation to separation.

$\text{OPT}^*(P, \Delta_P, c)$	
Input:	
<ul style="list-style-type: none"> • A well-described polytope $P \subseteq \mathbb{Q}^V$ with a separation oracle Δ_P, and • a linear objective $c \in \mathbb{Q}^V$. 	
Output:	
<ul style="list-style-type: none"> • $f = 1$ and $y = 0^V$, if P is unbounded along c; or otherwise • $f = 0$ and $y \in \mathbb{Q}^V$, s.t. if $P \neq \emptyset$ then $y \in P$ and $c^\top y = \max\{c^\top x \mid x \in P\}$. 	
<hr/> <pre> 1: $\sigma \leftarrow \text{REFINE}(0^V, c)$. 2: while true do 3: $(f, x) \leftarrow \text{OPT}([P]^\sigma, \Delta_{[P]^\sigma}, [c]^\sigma)$. 4: if aborted with σ' then 5: $\sigma \leftarrow \sigma'$. 6: else 7: $\sigma' \leftarrow \text{REFINE}(\sigma, \Delta_P([x]^{-\sigma}))$. 8: if $\sigma \neq \sigma'$ then 9: $\sigma \leftarrow \sigma'$. 10: else 11: return $(f, [x]^{-\sigma})$. </pre> <hr/> <pre> 12: oracle $\Delta_{[P]^\sigma}(x)$ 13: $d \leftarrow \Delta_P([x]^{-\sigma})$. 14: $\sigma' \leftarrow \text{REFINE}(\sigma, d)$. 15: if $\sigma \neq \sigma'$ then abort with σ'. 16: return $[d]^\sigma$. 17: end oracle </pre> <hr/>	

With this intuition in mind the formal proof is as follows.

Proof of Theorem 9. For completeness the entire algorithm OPT^* is described in Figure 4. The algorithm uses two subroutines REFINE and OPT . The subroutine $\text{REFINE}(\sigma, d)$ takes as input an index map σ of V represented in \mathbb{N}^V and a vector $d \in \mathbb{Q}^V$ and computes a new index map σ' with the following two properties:

- for all $v, v' \in V$ with $\sigma(v) < \sigma(v')$, $\sigma'(v) < \sigma'(v')$, and
- for all $v, v' \in V$ with $\sigma(v) = \sigma(v')$, $\sigma'(v) < \sigma'(v')$ iff $d_v < d_{v'}$.

It is straightforward to observe that when $\text{REFINE}(\sigma, d)$ produces an index map σ' which is different from σ , then σ' induces strictly more equivalence classes on V than σ does. Clearly, no index map can induce more than $|V|$ equivalence classes. The subroutine OPT solves the linear optimisation problem on an ordered space \mathbb{Q}^k with a given linear objective and a polytope given by a separation oracle. Without loss of generality assume OPT returns a integer-vector pair (f, y) which is $(1, 0^k)$ when the objective value is unbounded and

$(0, y)$ when $y \in \mathbb{Q}^k$ is an optimal point in the polytope if, and only if, the polytope is non-empty.

We first argue that the algorithm is correct, assuming the correctness of OPT and REFIN. For any index map $\sigma : V \rightarrow [k]$, $[P]^\sigma$ is a polytope by Proposition 13. We show that the procedure $\Delta_{[P]^\sigma}$ described in lines 12 to 17 acts as a separation oracle for $[P]^\sigma$ provided the answer given by the separation oracle Δ_P agrees with σ . If $\Delta_P([x]^{-\sigma})$ outputs $d = 0^V$, then this indicates that $[x]^{-\sigma} \in P$, and hence $x \in [P]^\sigma$ by Proposition 14. Trivially 0^V agrees with σ , so $[d]^\sigma = [0^V]^\sigma = 0^k$ is returned by $\Delta_{[P]^\sigma}$ correctly indicating that $x \in [P]^\sigma$. Otherwise, $d \neq 0^V$ and indicates that $[x]^{-\sigma} \notin P$ but $d^\top [x]^{-\sigma} > \max\{d^\top y \mid y \in P\}$. If d agrees with σ we have, by Proposition 12, $[d]^\sigma x > \max\{[d]^\sigma [y]^\sigma \mid y \in P\}$. This is equivalent to $[d]^\sigma x > \max\{[d]^\sigma y \mid y \in [P]^\sigma\}$. Hence $[d]^\sigma$ is the normal of a separating hyperplane of $[P]^\sigma$ through x . Since d agrees with σ , $\sigma' = \sigma$ and $[d]^\sigma$ is correctly returned. If d does not agree with σ , then REFIN produces a $\sigma' \neq \sigma$ and the procedure aborts. We conclude that (i) when $\Delta_{[P]^\sigma}$ does not abort it behaves as a separation oracle for $[P]^\sigma$, and (ii) when $\Delta_{[P]^\sigma}$ aborts the returned index map σ' is a strict refinement of σ . Thus $\Delta_{[P]^\sigma}$ is a separation oracle for $[P]^\sigma$, provided it does not abort. When OPT runs on $\Delta_{[P]^\sigma}$ without aborting the result must be a solution to the linear optimisation problem on $[P]^\sigma$.

Let $x \in [P]^\sigma$ be such that $[c]^\sigma x \geq \max\{[c]^\sigma y \mid y \in [P]^\sigma\}$, i.e., it is a solution to the linear optimisation problem on $[P]^\sigma$ along $[c]^\sigma$. By Proposition 14 this means that either (i) $[x]^{-\sigma} \in P$ or (ii) $\Delta_P([x]^{-\sigma})$ must disagree with σ . Applying Δ_P to $[x]^{-\sigma}$ distinguishes these two cases. In case (i), $[c]^\sigma x = c^\top [x]^{-\sigma} \geq \max\{c^\top y \mid y \in P\}$ by Proposition 12, because the initialisation of σ forces c to agree with σ . This means that $[x]^{-\sigma}$ is a solution to the linear optimisation problem for the polytope P and the objective c . In case (ii), $[x]^{-\sigma} \notin P$ but $\Delta_P([x]^{-\sigma})$ is guaranteed to improve the index map. In the case that the linear optimisation algorithm returns that $[P]^\sigma$ is unbounded in the direction of $[c]^\sigma$, it implies, via similar analysis, that P is unbounded in the direction c . Finally, when the optimisation algorithm reports that $[P]^\sigma$ is empty we conclude that P must be empty as well, because if P contains at least one point then $[P]^\sigma$ must also contain at least one point. The algorithm correctly translates the solutions for the linear optimisation problem for $[P]^\sigma$ back to solutions for P . This means that when OPT* returns its result is correct.

We now observe that this algorithm runs in polynomial time. The main loop cannot execute more than $|V|$ times, because, as established above, at each step either the index map σ is improved to induce more equivalence classes—up to $|V|$ classes—or the algorithm returns a correct solution to the linear optimisation problem on P . The size of all of the objects referred to by the algorithm can be polynomially bounded by a function of the input length. In particular, since P is well-described by Δ_P , there is a polynomial bound on its bit complexity and this induces a bound on the size of $[P]^\sigma$ through Proposition 13 and implies that $[P]^\sigma$ is well described. This implies that the bit complexity of values in the algorithm can be bounded by some fixed polynomial. This means that folding and unfolding can be computed in polynomial time. Similarly, a naive implementation of the subroutine REFIN can be seen to run in polynomial time in $|V|$ and the bit complexity of its input rational vector. Since $[P]^\sigma$ is a

well-described polytope with a polynomial-time separation oracle $\Delta_{[P]^\sigma}$ we can use the polynomial-time algorithm for OPT from Theorem 8 to solve the linear optimisation problem on $[P]^\sigma$. Combining all these parts implies that OPT* is a polynomial-time algorithm.

We conclude by arguing that the behavior of OPT* can be simulated in FPC. Relative to an index map σ expressible in FPC, folding and unfolding can be expressed in FPC using basic rational arithmetic. It is similarly routine to express REFINe in FPC by defining the equivalence classes and then counting sizes to determine the correct position of each equivalence class relative to an FPC-definable σ and vector. Moreover, there is a FPC-interpretation $\Sigma_{[P]^\sigma}$ expressing the separation problem for $[P]^\sigma$. This implies there is an FPC-interpretation for the combination of OPT and the separation oracle given by $\Sigma_{[P]^\sigma}$, because the polytope $[P]^\sigma$ lies in an ordered space and the Immerman-Vardi theorem [Var82, Imm86] indicates that any polynomial-time property of ordered structures can be defined in fixed-point logic (and hence in FPC). It is easy to see that the algorithm's main loop and control structure can be simulated in FPC. Combining everything gives an FPC-interpretation simulating OPT* and hence expressing the linear optimisation problem for P given a FPC-interpretation expressing the separation problem for P . \square

In the next four sections we demonstrate a number of applications of Theorem 9 for expressing classical combinatorial optimisation problems in fixed-point logic with counting.

5 Application: Maximum Flow

Let $G = (V, c)$ be a graph with non-negative edge capacities, that is, $c : V \times V \rightarrow \mathbb{Q}_{\geq 0}$. For a pair of distinct vertices $s, t \in V$ an (s, t) -flow is a function $f : V \times V \rightarrow \mathbb{Q}_{\geq 0}$ satisfying capacity constraints $0 \leq f(u, v) \leq c(u, v)$ on each pair of distinct $u, v \in V$ and conservation constraints $\sum_{v \in V} (f(v, u) - f(u, v)) = 0$ on all vertices $u \in V \setminus \{s, t\}$. The *value* $\text{val}(f)$ of the flow f is simply the difference in in-flow and out-flow at t , i.e., $\sum_{v \in V} (f(v, t) - f(t, v))$. Observe that any flow f can be *normalised* to f' so that for any pair of distinct $u, v \in V$ at least one of $f'(u, v)$ and $f'(v, u)$ is zero (i.e., if $f(u, v) \geq f(v, u)$, set $f'(u, v) := f(u, v) - f(v, u)$ and $f'(v, u) := 0$; obviously this preserves the capacity constraints, the conservation constraints and the value of the flow). A *maximum (s, t) -flow* of G is a flow whose value is maximum over all (s, t) -flows.

Observation 15. Fix $G = (V, c)$ and $s, t \in V$. Let f_1, f_2 be two (s, t) -flows in G . Fix any $\alpha \in \mathbb{Q}$ with $0 \leq \alpha \leq 1$, let $f' := \alpha \cdot f_1 + (1 - \alpha) \cdot f_2$. Then f' is an (s, t) -flow of G and $\text{val}(f') = \alpha \cdot \text{val}(f_1) + (1 - \alpha) \cdot \text{val}(f_2)$. In particular, if f_1 and f_2 are maximum (s, t) -flows then so is any convex combination f' .

Let $G|_f := (V, c - f)$ denote the *residual graph* of G with respect to the flow f . The standard formulation of the maximum (s, t) -flow problem as a linear

program is as follows:

$$\begin{aligned}
& \max \sum_{v \in V} (f(v, t) - f(t, v)) && \text{subject to} \\
& \sum_{v \in V} (f(v, u) - f(u, v)) = 0, \quad \forall u \in V \setminus \{s, t\} \\
& 0 \leq f(u, v) \leq c(u, v), \quad \forall u \neq v \in V.
\end{aligned} \tag{2}$$

5.1 Expressing Maximum Flow in FPC

Observe that there are $|V|(|V| - 1)$ variables in linear program (2) corresponding to $f(u, v)$ for distinct $u, v \in V$. The program has $2|V|^2 - 4$ constraints. Both the variables and constraints can be indexed by tuples of elements from V . It can easily be established that the maximum (s, t) -flow linear program can be defined by an FPC interpretation. That is to say, suppose that a capacitated graph (V, c) is given as a τ_{mat} -structure with universe V where the rational matrix $c \in \mathbb{Q}_{\geq 0}^{V \times V}$ codes the capacities. Then, there is an FPC interpretation from τ_{mat} to $\tau_{\text{mat}} \uplus \tau_{\text{vec}}$ that takes a capacitated graph (V, c) and a pair $s, t \in V$ and explicitly expresses a constraint matrix A and vector b encoding the corresponding flow polytope. The flow polytope is bounded because each variable is constrained from both above and below. Further the flow polytope is nonempty because the capacities in G are nonnegative and hence the zero flow is a member of the polytope. Thus, because this polytope is explicit, Theorem 10 immediately gives an FPC interpretation expressing the optimisation problem on the flow polytope.

Theorem 16. *There is an FPC interpretation $\Phi(s, t)$ of τ_{mat} in τ_{mat} which takes a τ_{mat} -structure coding a capacitated graph G to a τ_{mat} -structure coding a maximum (s, t) -flow of G .*

Note that as the interpretation Φ defines a particular flow, the flow must, in some sense, be canonical because it is produced without making any choices. Informally, it is a convex combination of maximum flows resulting from the consideration of all orderings consistent with the most refined index map determined by the FPC interpretation of Theorem 9. This is possible because of Observation 15. In our remaining applications—minimum cut and maximum matching—the analog of Observation 15 does not hold: Convex combinations of cuts or matchings are not necessarily cuts or matchings. In the former it is still possible to define the notion of a canonical optimum. In the latter case it is easy to observe, as noted in the introduction, that defining a canonical maximum matching is not possible.

6 Application: Minimum Cut

An (s, t) -cut of a capacitated graph $G = (V, c)$ is a subset C of the vertices V which contains s but not t . The *value* $\text{val}(C)$ of the cut C is the sum of the capacity of edges going from vertices in C to vertices in $V \setminus C$. A *minimum (s, t) -cut* of G is a cut whose value is the minimum over all (s, t) -cuts. A *minimum cut* of G is a minimum (s, t) -cut over all choices of distinct vertices s, t . By

the max-flow/min-cut theorem, a maximum (s, t) -flow and a minimum (s, t) -cut have the same value. This duality allows the construction of minimum cuts from maximum flows. In this section we describe an FPC formula defining a minimum (s, t) -cut in a graph using the FPC interpretation for the maximum (s, t) -flow problem given by Theorem 16; we show this minimum cut is canonical in a strong sense.

6.1 Expressing Canonical Minimum Cut in FPC

First, we define a notion of directed reachability in capacitated graphs. A vertex v is *reachable* from a vertex u if there is a path in the graph which follows directed edges with non-zero capacity (this is exactly directed reachability in the graph induced by eliminating zero capacity edges). Let f be a maximum (s, t) -flow in $G = (V, c)$ with normalised flow f' . Define $C_f := \{v \in V \mid v \text{ reachable from } s \text{ in } G|_{f'}\}$. C_f is a minimum (s, t) -cut in G . Since f' is normalised, every edge leaving C_f must be at full capacity in f' .

Given the FPC interpretation Φ from Theorem 16 expressing an (s, t) -flow f , it is not difficult to construct a formula of FPC which defines the normalised flow f' and then the set of vertices C_f .

Theorem 17. *There is a formula $\xi(x, s, t)$ of FPC which given a τ_{mat} -structure coding a capacitated graph $G = (V, c)$, defines the vertices in a minimum (s, t) -cut of G .*

In fact, the cut C_f does not depend on f at all, as we show next. Indeed, C_f is the smallest minimum (s, t) -cut in the sense that it is contained in all other minimum (s, t) -cuts of G .

Lemma 18. *Let $G = (V, c)$ be a capacitated graph with distinct vertices $s, t \in V$. Then the cut C_f is independent of the choice of a maximum (s, t) -flow f of G . Moreover, C_f is the intersection of all minimum (s, t) -cuts of G .*

Proof of Lemma 18. Suppose not. There are two distinct minimum (s, t) -cuts $C := C_f$ and $C' := C_{f'}$ with corresponding normalised (s, t) -flows f and f' . Since C and C' are different there exists, without loss of generality, $v \in C' \setminus C$. Consider the flows through $\overline{C} \cap C'$. We use a, a', b, b', c, c' to denote the net flows into and out of this set. See Figure 5 for definitions. By definition of C and C' there is no flow in f from \overline{C} to C nor is there flow in f' from \overline{C}' to C' as otherwise vertices in the complementary cuts would be reachable from s .

The flow conservation constraints require the flow into $\overline{C} \cap C'$ be matched by the outflow in both f and f' . This implies that $a + b = c$ and $a' = b' + c'$. In addition $a \geq a'$ and $c' \geq c$, because these edges must be at full capacity in f and f' respectively. Combining these equalities and inequalities produces $a + b \leq c'$ and $b' + c' \leq a$. Adding these two constraints together gives $a + b + b' + c' \leq a + c'$. Since all values are non-negative we have $b = b' = 0$. This implies $a = c$ and $a' = c'$. Then, reusing $a \geq a'$ and $c' \geq c$ we conclude $a = c = a' = c'$. This means that in f' the edges going from $C \cap C'$ to $\overline{C} \cap C'$ are at full capacity, and thus no vertex in $\overline{C} \cap C'$ is reachable from s in flow f' . This is a contradiction.

Since the flow from between $C \cap C'$ and $\overline{C} \cap C'$ is the same in both f and f' , flow f' witnesses that $C \cap C'$ is a minimum (s, t) -cut of G . This implies the “moreover” part of the statement and completes the proof. \square

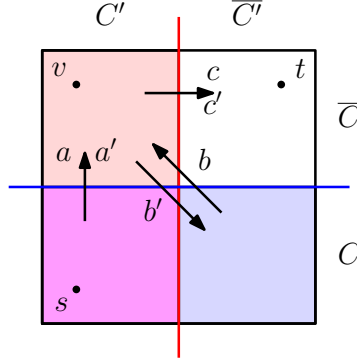


Figure 5: Diagram for the proof of Lemma 18. Here the variables indicate the net flow between two sets under flows f and f' .

Note that this proof is similar to the “lemma on a quadrangle” from [DKL76], but that proof does not immediately go through because $\overline{C} \cap C'$ may not be a (s, t) -cut.

Lemma 18 implies that the FPC formula ξ of Theorem 17 defines a *unique* (s, t) -cut of the graph G and for this reason we call it the *canonical* minimum (s, t) -cut of G : $K_{G,s,t}$.

7 Application: Minimum Odd Cut

The *minimum odd cut* problem is closely related to the minimum (s, t) -cut problem. Here the goal is to define a minimum odd cut of a graph G . That is, a cut of odd size whose value is minimum among all odd size cuts of G . A capacitated graph $G = (V, c)$ is *symmetric* if for all $u, v \in V$, $c(u, v) = c(v, u)$. Observe that in symmetric graphs the set C is a minimum (s, t) -cut iff its complement $\overline{C} := V \setminus C$ is a minimum (t, s) -cut. In this section we prove that in each symmetric graph G there is at least one pair of vertices s, t such that the canonical minimum (s, t) -cut $K_{G,s,t}$ is a minimum odd cut of G . The results and techniques discussed in this section are entirely graph theoretic.

Before continuing we must define several special types of cuts. We extend a capacitated graph $G = (V, c)$ to a *marked* capacitated graph $G' = (V, c, M)$ with a marking $M \subseteq V$. We call a vertex $v \in V$ *marked* if $v \in M$. A cut C of a marked graph G is said to be a *marked cut*, if both C and \overline{C} contain a marked vertex. A cut C of a graph $G = (V, c)$ with $|V|$ even is said to be an *odd* cut if $|C|$ is odd. A marked cut C of a marked graph $G = (V, c, M)$ with $|M|$ even is said to be an *odd marked* cut if $|C \cap M|$ is odd (note that this corresponds to the simpler notion when $M = V$). For any set \mathcal{C} of cuts we define the *basic* cuts in \mathcal{C} to be $\{C \in \mathcal{C} \mid \forall C' \in \mathcal{C}, C = C' \text{ or } C \not\supseteq C'\}$. Note that if \mathcal{C} is non-empty it must contain at least one basic cut. When \mathcal{C} is the set of minimum (s, t) -cuts, the formula ξ of Theorem 17 defines the unique basic cut in \mathcal{C} . We frequently describe sets of cuts by a sequence of the above adjectives and determine meaning by first evaluating the adjective which appear closest to the word “cut”. The most complex cuts we consider are “basic minimum odd marked cuts”.

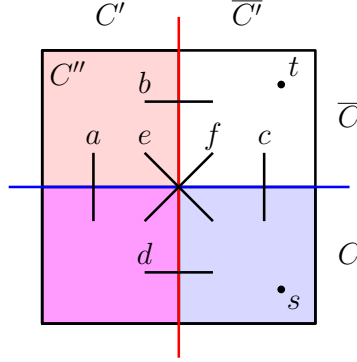


Figure 6: Diagram for the proof of Lemma 19.

Section 7.1 develops several technical properties of cuts of marked symmetric graphs. Using these properties Section 7.2 shows that there is a canonical minimum (s, t) -cut of a symmetric graph G which is also a minimum odd cut of G .

7.1 Intersections of Minimum Cuts

We now prove two technical properties involving the intersections of marked cuts. The first says that basic minimum marked cuts do not have complicated intersections with basic minimum (s, t) -cuts. The proof is similar in spirit to the proof of Lemma 18

Lemma 19. *Let $G = (V, c, M)$ be a marked symmetric graph. Let $s, t \in M$ be distinct vertices and let C be a basic minimum (s, t) -cut of G . For every basic minimum marked cut C' of G one of the following holds: (i) $C \supseteq C'$, (ii) $C \cap C' = \emptyset$ or (iii) $\{s, t\} \cap C' \neq \emptyset$.*

Proof. Fix any basic minimum marked cut C' of G . Suppose neither property (ii) or (iii) holds; it suffices to show that property (i) holds. Thus our goal is to show that $C'' := \overline{C} \cap C' = \emptyset$ assuming that $C \cap C' \neq \emptyset$ and $\{s, t\} \cap C' = \emptyset$. See Figure 6 for a diagram of the general configuration of these cuts and for the definitions of variables labelling the symmetric capacity crossing between the various sets.

Observe that $C \cap \overline{C'}$ is an (s, t) -cut, however, it cannot be a minimum (s, t) -cut because $C \supsetneq C \cap \overline{C'}$ (since $C \cap C' \neq \emptyset$) and C is a basic minimum (s, t) -cut. Thus

$$c + d + e = \text{val}(C \cap \overline{C'}) > \text{val}(C) = a + c + e + f$$

and hence $d > a + f$.

Since C' is a marked cut of G , C' contains a marked vertex. Suppose that C'' contains a marked vertex. In this case C'' is marked cut of G , because it contains at least one marked vertex, but not all marked vertices (e.g., s). Since C' is a basic minimum marked cut of G and $C'' \subsetneq C'$, C'' cannot be a minimum marked cut of G . This implies that

$$a + b + e = \text{val}(C'') > \text{val}(C') = b + d + e + f$$

and hence that $a > d + f$. Combining this with the inequality $d > a + f$ derived from C being a basic minimum (s, t) -cut we have $d > a + f > d + 2f$ which is a contradiction because all the edges have non-negative capacity. Thus C'' cannot contain a marked vertex. This implies that $C \cap C'$ contains a marked vertex and is hence a marked cut of G .

Suppose that $C'' \neq \emptyset$. Since C' is a basic minimum marked cut of G , $C \cap C' \subsetneq C'$ cannot be a minimum marked cut of G . Thus

$$a + d + f = \text{val}(C \cap C') > \text{val}(C') = b + d + e + f$$

and hence $a > b + e$. Combining this with the value of C and the fact that the edges are non-negative implies

$$\text{val}(C) = a + c + e + f > b + c + 2e + f \geq b + c + f = \text{val}(\overline{C} \cap \overline{C'}).$$

Thus $\text{val}(C) > \text{val}(\overline{C} \cap \overline{C'})$. As $\overline{C} \cap \overline{C'}$ contains t but not s , it is an (t, s) -cut. Moreover, $\overline{C} \cap \overline{C'}$ is an (t, s) -cut with value strictly less than that of C which is a contradiction because C is a minimum (s, t) -cut and the capacities are symmetric. Therefore $C'' = \emptyset$ and the proof is complete. \square

The second lemma says that given a basic minimum odd marked cut C , there exists a minimum marked cut C' which does not have a complicated intersection with C . The proof is quite similar to those of Lemmas 18 & 19.

Lemma 20. *Let $G = (V, c, M)$ be a marked symmetric graph with $|M|$ even. Let C be a basic minimum odd marked cut of G . There exists a minimum marked cut C' of G such that one of the following holds: (i) $C \supseteq C'$ or (ii) $C \cap C' = \emptyset$.*

Proof. Fix any minimum marked cut C' of G . The complementary cut $\overline{C'}$ of C' is also a minimum marked cut of G because G is symmetric. If $C \supseteq C'$ or $C \cap C' = \emptyset$ the condition is immediately satisfied. If $C' \supseteq C$, then $\overline{C'}$ has no intersection with C and the minimum marked cut $\overline{C'}$ satisfies (ii). If $\overline{C} \cap \overline{C'} = \emptyset$, then $C \supseteq \overline{C'}$ and hence minimum marked cut $\overline{C'}$ satisfies (i). In the case that none of these things happen we observe that $C \cap C'$, $C \cap \overline{C'}$, $\overline{C} \cap C'$ and $\overline{C} \cap \overline{C'}$ are all non-empty. Because C is an odd marked cut there are two disjoint (but symmetric) cases:

1. $C \cap C'$ is an odd marked cut.

See Figure 7 for a diagram of this case and for the definitions of variables labelling the edges crossing between the various sets. There are two further sub-cases:

1.a. $\overline{C} \cap \overline{C'}$ contains a marked vertex.

Since C' is a marked cut of G , C' contains a marked vertex. Since $\overline{C} \cap \overline{C'}$ also contains a marked vertex, $\overline{C} \cap \overline{C'}$ is a marked cut of G . Furthermore, C' is a minimum marked cut of G and hence

$$b + d + e + f = \text{val}(C') \leq \text{val}(\overline{C} \cap \overline{C'}) = b + c + f.$$

We resolve that $c \geq d + e$. Similarly, since C is basic, the odd cut $C \cap C' \subsetneq C$ of G must have a larger value than C , and thus $d > c + e$. Combining the two inequalities we conclude $d > c + e \geq d + 2e$. This is a contradiction because values are non-negative.

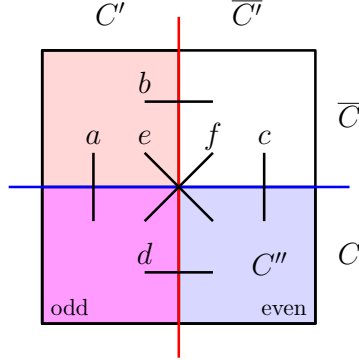


Figure 7: Diagram for the proof of Lemma 20.

1.b. $\overline{C} \cap \overline{C}'$ contains no marked vertices.

Since \overline{C}' is a marked cut, it contains a marked vertex. This implies that $C'' := C \cap \overline{C}'$ contains a marked vertex, and hence that C'' is a marked cut of G . Thus to satisfy (i) it suffices to show that C'' is a *minimum* marked cut of G .

Because C is an odd marked cut and $C \cap C'$ is an odd marked cut, C'' contains an even number of marked vertices. Since $\overline{C} \cap \overline{C}'$ contains no marked vertices, \overline{C}' contains an even number of marked vertices. This implies that $\overline{C} \cap C'$ is an odd marked cut of G . Since C is a minimum odd marked cut of G and $\overline{C} \cap C'$ is an odd marked cut we have that

$$a + c + e + f = \text{val}(C) \leq \text{val}(\overline{C} \cap C') = a + b + e.$$

This implies that $b \geq c + f$. Similarly, since C' is a minimum marked cut of G and C'' is a marked cut of G we have that $c \geq b + f$. Combining these two inequalities $b \geq c + f \geq b + e + f$. As all values are non-negative we must conclude that $b = c$ and $e = f = 0$. This means that the value of C'' is

$$\text{val}(C'') = c + d + f = b + d + e + f = \text{val}(C')$$

and we conclude that C'' is a minimum marked cut of G .

2. $C \cap \overline{C}'$ is an odd marked cut.

Repeat Case 1 with C' and \overline{C}' swapped. □

7.2 Some Canonical Min (s, t) -Cuts are Min Odd Cuts

In this subsection we show that for any symmetric graph $G = (V, c)$ there exists a pair of vertices $s, t \in V$ such that the canonical minimum (s, t) -cut $K_{G, s, t}$ is a minimum odd cut of G .

The intuition for the proof is as follows. Let G be a symmetric graph with minimum odd cut C . Mark all vertices in G . We use Lemma 20 to locate a basic minimum marked cut D of G which C does not partition (that is, D is either contained in C or disjoint from C). If D is an odd cut, and because D is basic, a canonical minimum (s, t) -cut separating a vertex $s \in D$ from a vertex $t \notin D$ is also a minimum odd cut of G and we are done. Otherwise $|D|$ is even

Figure 8: A subroutine to collapse a set of vertices in a graph to a new single vertex.

COLLAPSE(G, D, z)

Input: A marked capacitated graph $G = (V, c, M)$, a set $D \subseteq V$ and $z \notin V$.

Output: The graph obtained from G by collapsing of the vertices in D to z .

- 1: $V' \leftarrow V(z/D)$.
 - 2: $c'(u, w) \leftarrow \begin{cases} c(u, w), & u, w \in V \setminus D, \\ \sum_{x \in D} c(u, x), & u \in V \setminus D, w = z, \\ \sum_{x \in D} c(x, w), & w \in V \setminus D, u = z. \end{cases}$
 - 3: $M' \leftarrow M \setminus D$.
 - 4: **return** (V', c', M') .
-

and we form a new graph G' by collapsing D into a new super-vertex z , and then setting the effected capacities so that value of cuts which do not partition D are unchanged. Since C does not partition D and $|D|$ is even, the collapsed version C' of C is a minimum odd marked cut of G' . We repeat this collapsing procedure maintaining a graph G' and minimum odd marked cut C' until we locate a minimum marked cut D of G' that is also minimum odd marked cut. This provides marked vertices s, t such that the canonical minimum (s, t) -cut of G' is a minimum odd cut of G' . The proof concludes using Lemma 19 to translate this fact back to the original graph G and in doing so argues that the canonical minimum (s, t) -cut of G is a minimum odd cut.

We now formalise the notion of collapsing a graph. We begin by establishing notation for substituting sets into sets. Let $C, D \subseteq V$ and $z \notin V$ such that either $C \supseteq D$ or $C \cap D = \emptyset$. Define $C(z/D)$ to be a subset of $V' := (V \setminus D) \cup \{z\}$

$$C(z/D) := \begin{cases} (C \setminus D) \cup \{z\}, & C \supseteq D, \\ C, & C \cap D = \emptyset, \end{cases}$$

and for $C' \subseteq V'$ define $C'(D/z)$ to be a subset of V

$$C'(D/z) := \begin{cases} (C' \setminus \{z\}) \cup D, & z \in C', \\ C', & z \notin C'. \end{cases}$$

Observe that $(C(z/D))(D/z) = C$.

The subroutine COLLAPSE(G, D, z) in Fig. 8 describes a method of collapsing the vertex set $D \subseteq V$ in the marked graph $G = (V, c, M)$ to a single new super-vertex z . This subroutine is designed to preserve a basic minimum odd marked cut C with respect to a well-chosen marked cut D .

Lemma 21. *Let $G = (V, c, M)$ be a marked symmetric graph with $|M|$ even. Let $z \notin V$ and $C, D \subseteq V$ be marked cuts of G such that $C \supseteq D$ or $C \cap D = \emptyset$. Define $C' := C(z/D)$ and $G' := (V', c', M') := \text{COLLAPSE}(G, D, z)$.*

1. *The value of C in G is identical to the value of C' in G' .*
2. *If C is a basic minimum odd marked cut of G and $|D \cap M|$ is even, then C' is a basic minimum odd marked cut of G' , and $|M'|$ is even, non-zero and $M' \subsetneq M$.*

Proof. Suppose $C \supseteq D$. The definition of COLLAPSE implies that

$$\begin{aligned} \text{val}(C) &= \sum_{u \in C, v \in \overline{C}} c(u, v) = \sum_{u \in C \setminus D, v \in \overline{C}} c(u, v) + \sum_{u \in D, v \in \overline{C}} c(u, v) \\ &= \sum_{u \in C \setminus D, v \in \overline{C}} c(u, v) + \sum_{v \in \overline{C}} c(z, v) \\ &= \sum_{u \in (C \setminus D) \cup \{z\}, v \in \overline{C}} c(u, v) = \text{val}(C'). \end{aligned}$$

The case of $C \cap D = \emptyset$ is analogous. We conclude property 1 holds.

Assume the hypothesis of property 2. Observe that D contains an even number of the marked vertices M but not all of them because D is a marked cut of G . The subroutine COLLAPSE sets $M' = M \setminus D$. Therefore $|M'|$ is even because $|D \cap M|$ and $|M|$ are even, and thus M' meets the required conditions.

We also determine that C' is an odd marked cut of G' , because $|D \cap M|$ is even and C is an odd marked cut which either contains D or is disjoint from D .

Suppose C' is not a *minimum* odd marked cut of G' . Then there exists an odd marked cut C'' of G' with smaller value than C' . The cut $C''(D/z)$ is a marked cut of G which does not partition D . By property 1 $\text{val}(C''(D/z)) = \text{val}(C'')$ and $\text{val}(C) = \text{val}(C(z/D)) = \text{val}(C')$. Hence

$$\text{val}(C''(D/z)) = \text{val}(C'') < \text{val}(C') = \text{val}(C(z/D)) = \text{val}(C).$$

However, $C''(D/z)$ is an odd marked cut because C' is an odd marked cut, z is not marked and D contains an even number of marked vertices. This means that C is not a minimum odd marked cut of G which contradicts the hypothesis. Therefore C' is a *minimum* odd marked cut of G' .

Similarly, suppose C' is not a *basic* minimum odd marked cut of G' . Then there exists a minimum odd marked cut C'' of G' with $C'' \subsetneq C'$. By property 1, it follows that $C''(D/z)$ is a minimum odd marked cut of G with $C''(D/z) \subsetneq C$, this contradicts the basicness of C . Therefore C' is a *basic* minimum odd marked cut of G' and the proof is complete. \square

With the key properties of COLLAPSE established we are ready to prove the main result of this section.

Theorem 22. *Let $G = (V, c, M)$ be a marked symmetric graph with even $|M| > 0$. There exist $s, t \in M$ such that the canonical minimum (s, t) -cut of G is a minimum odd marked cut of G .*

Proof. Fix a basic minimum odd marked cut C of G . Figure 9 describes the algorithm WITMINODDCUT that computes vertices s and t witnessing the claim of the theorem from C by iteratively collapsing G . To prove the theorem it suffices to argue the algorithm halts and produces $(s, t) \in V^2$ such that the canonical minimum (s, t) -cut $K_{G, s, t}$ is a minimum odd marked cut.

As the algorithm runs it maintains the invariant that C^i is a basic minimum odd marked cut of G^i . Observe that this is initially true for $C^0 = C$ because $M^0 = M$ and C is a basic minimum odd marked cut of G . Suppose C^i is a basic minimum odd marked cut of G^i . The basic minimum marked cut D^i of G^i with

Figure 9: An algorithm producing a witness for a minimum odd marked cut.

WITMINODDCUT(G, C)

Input: A symmetric graph $G = (V, c, M)$ and a minimum odd marked cut C of G .

Output: $(s, t) \in M^2$ such that $K_{G,s,t}$ is a minimum odd marked cut.

```

1:  $i \leftarrow 0$ .
2:  $G^0 := (V^0, c^0, M^0) \leftarrow (V, c, M)$ .
3:  $C^0 \leftarrow C$ .
4: while true do
5:   Let  $D^i$  be a basic minimum marked cut of  $G^i$  such that  $C^i \supseteq D^i$  or
      $C^i \cap D^i = \emptyset$ .
6:   if  $D^i$  is an odd marked cut of  $G^i$  then
7:     return  $(s, t)$  with  $s \in D^i \cap M^i$  and  $t \in M^i \setminus D^i$ .
8:    $G^{i+1} \leftarrow \text{COLLAPSE}(G^i, D^i, z^i)$ .
9:    $C^{i+1} \leftarrow C^i(z^i/D^i)$ .
10:   $i \leftarrow i + 1$ .

```

$C^i \supseteq D^i$ or $C^i \cap D^i = \emptyset$ is guaranteed to exist by Lemma 20 (if the cut given by that lemma is not basic there must be a basic minimum marked cut strictly within it that continues to satisfy the intersection properties with C^i). If D^i is an odd marked cut, the algorithm halts at line 7. Otherwise the graph G^i and cut C^i are collapsed relative to D^i . The second property of Lemma 21 implies that C^{i+1} is a basic minimum odd marked cut of G^{i+1} . Thus the invariant holds.

Lemma 21 also implies that as the algorithm runs, $|M^i|$ is even and $M^{i+1} \subsetneq M^i$. The invariant and $M^0 = V$ imply that the test in line 6 will be successful and cause the algorithm to halt within $\frac{|V|}{2}$ iterations. Because D^i is a marked cut of G^i , when line 7 is reached $D^i \cap M^i$ and $M^i \setminus D^i$ are non-empty disjoint sets. This means that distinct s and t exist and are returned by the algorithm. Let r be the value of i when the algorithm halts. Fix any $s \in D^r \cap M^r$ and $t \in D^r \setminus M^r$. We use the shorthand K^i to denote the canonical minimum (s, t) -cut $K_{G^i,s,t}$ for $0 \leq i \leq r$. It remains to argue that $K^0 = K_{G,s,t}$ is a minimum odd marked cut.

Since the cut D^r is a minimum marked cut of G^r and $s, t \in M^r$, D^r is also a minimum (s, t) -cut of G^r and it has the same value as the canonical minimum (s, t) -cut K^r . This implies that K^r is a minimum marked cut of G^r because $s, t \in M^r$. By Lemma 18, D^r contains K^r , but D^r is also basic, so we conclude that $D^r = K^r$.

The first property of Lemma 21 implies that $\text{val}(C^i(z^i/D^i)) = \text{val}(C^{i+1})$ for all $0 \leq i < r$, and hence that $\text{val}(C^i) = \text{val}(C)$ for all $0 \leq i \leq r$. Since K^r is a minimum marked cut of G^r and C^r is a marked cut of G^r , $\text{val}(K^r) \leq \text{val}(C^r) = \text{val}(C)$. The first part of Lemma 21 also implies that $\text{val}(K^{i+1}(D^i/z^i)) = \text{val}(K^{i+1})$ for all $0 \leq i < r$. Since $K^{i+1}(D^i/z^i)$ is an (s, t) -cut of G^i , $\text{val}(K^i) \leq \text{val}(K^{i+1}(D^i/z^i))$ for all $0 \leq i < r$. Hence we conclude that $\text{val}(K^0) \leq \text{val}(K^r) \leq \text{val}(C)$.

It remains to argue that K^0 is an odd marked cut. Since K^r is an odd marked cut, it suffices to show that K^i is an odd marked cut if K^{i+1} is an odd

marked cut, for all $0 \leq i < r$. To this end assume that K^{i+1} is an odd marked cut. Apply Lemma 19 with G^i , K^i and D^i ; we note that (i) $K^i \supseteq D^i$, (ii) $K^i \cap D^i = \emptyset$, or (iii) $\{s, t\} \cap D^i \neq \emptyset$. Property (iii) cannot hold because s and t are selected after D^i was collapsed. This means that K^i either contains all of D^i or is disjoint from D^i . Because the K^i and K^{i+1} are canonical (s, t) -cuts,

$$K^i \subseteq K^{i+1}(D^i/z^i) \subseteq (K^i(z^i/D^i))(D^i/z^i) = K^i.$$

As the algorithm did not halt at step i , $|D^i \cap M^i|$ is even and thus K^i is an odd marked (s, t) -cut.

We conclude that K^0 is an odd marked cut with value at most that of a minimum odd marked cut C of G . Therefore $K^0 = K_{G,s,t}$ is a minimum odd marked cut. \square

Consider the following procedure for locating a set of minimum odd marked cuts in a marked symmetric graph $G = (V, c, M)$: For all distinct $s, t \in M$ compute the canonical minimum (s, t) -cut $K_{G,s,t}$, eliminate those cuts which are not odd, then eliminate those cuts which are not minimal. Theorem 22 indicates that some cuts remain and that those cuts are minimum odd cuts of G . Note that the algorithm WITMINODDCUT in the proof of Theorem 22 is used only in the analysis and not actually run during the above procedure. This simple procedure for defining a non-empty set of minimum odd cuts is critical to expressing the separation problem for the matching polytope in FPC.

8 Application: Maximum Matching

Let $G = (V, E)$ be an undirected graph. A *matching* $M \subseteq E$ is defined by the property that no two edges in M are incident to the same vertex. A matching M is *maximum* if no matchings with size larger than M exist. A maximum matching is *perfect* if every vertex in G is incident to some edge in the matching (i.e., $|M| = \frac{|V|}{2}$).

8.1 Maximum Matching Program

Maximum matching has an elegant representation as a linear program. In fact, it is an instance of a slightly more general problem: b -matching. Let $c \in \mathbb{Q}_{\geq 0}^E$, $b \in \mathbb{N}^V$ and $A \in \{0, 1\}^{V \times E}$ be the incidence matrix of the undirected graph $G = (V, E)$: the columns of A correspond to the edges E and the rows to the vertices V , and $A_{ve} = 1$ if edge e is incident on vertex v . Alternatively we view edges $e \in E$ as two-element subsets of V . The goal of the b -matching problem is to determine an optimum of the following *integer* linear program

$$\max c^\top y \quad \text{subject to} \quad Ay \leq b, y \geq 0^E. \quad (3)$$

We obtain the usual maximum matching problem in the special case where $b = 1^V$ and $c = 1^E$.

Generically, integer programming is NP-complete, so instead of trying to directly solve the above program we consider the following relaxation as a rational

linear program.

$$\begin{aligned}
& \max c^\top y && \text{subject to} \\
& Ay \leq b, \\
& y \geq 0^E, \\
& y(W) \leq \frac{1}{2}(b(W) - 1), \quad \forall W \subseteq V \text{ with } b(W) \text{ odd},
\end{aligned} \tag{4}$$

where $y(W) := \sum_{e \in E, e \subseteq W} y_e$ and $b(W) := \sum_{v \in W} b_v$. Here we have added a new set of constraints over subsets of the vertices. The integral points which satisfy (3), also satisfy the additional constraints that are added in (4). To see this, let y be a feasible integral solution, consider some set W with $b(W)$ odd. If $|W| = 1$, then $y(W) = 0$ because no edges have both endpoints in W , so assume $|W| \geq 2$. It follows that $2y(W) \leq b(W)$, by summing the constraints of $Ay \leq b$ over W with respect to only the edges with *both* endpoints in W . Since $b(W)$ is odd, $\frac{1}{2}b(W)$ is half integral, but $y(W)$ is integral because y is an integral solution; this means the constraint $y(W) \leq \frac{1}{2}(b(W) - 1)$ is a valid constraint for all integral solutions. In fact [Edm65] shows something stronger.

Lemma 23 ([Edm65, Theorem P]). *The extremal points of the linear program (4) are integral and are the extremal solutions to the b -matching problem.*

Thus to solve b -matching it suffices to solve the relaxed linear program (4). As mentioned before, it will not be possible to show that FPC can generally define a particular maximum matching, there can be simply too many. However, the above lemma means that the existence of a (likely non-integral) feasible point y of (4) with value $c^\top y$ witnesses the existence of a maximum b -matching with value at least $c^\top y$. In addition, the number of constraints in this linear program is exponential in the size of the graph G . Thus, we cannot hope to interpret this linear program directly in G , using FPC. Rather what we can show is that there is an FPC interpretation which, given G , b and c , expresses the separation problem for the b -matching polytope in the linear program (4). Combining this with Theorem 9 gives an FPC interpretation expressing the b -matching optimum.

8.2 Expressing Maximum Matching in FPC

The b -matching polytopes have a natural representation over $\tau_{\text{match}} := \tau_{\text{mat}} \uplus \tau_{\text{vec}}$. Although the number of constraints in the b -matching polytope may be large, the individual constraints have size at most a polynomial in the size of the matching instance. Thus this representation is well-described.

We now describe an FPC interpretation expressing the separation problem for the b -matching polytope given a τ_{match} -structure coding the matrix A and bound vector b . As in the explicit constraint setting, our approach is to come up with a definable set of violated constraints iff the candidate point is infeasible. We then define a canonical violated constraint by summing this definable violated set. Identifying violated vertex and edge constraints can easily be done in FPC as before. However, it is not immediately clear how to do this for the odd set constraints.

To overcome this hurdle we follow the approach of [PR82]. Let y be point which we wish to separate from the matching polytope. Define $s := b - Ay$ to

be the slack in the constraints $Ay \leq b$. Analogous to $b(W)$, define $s(W) := \sum_{v \in W} s_v$. Observe that $2y(W) + y(W : V \setminus W) + s(W) = b(W)$ (here $y(W : V \setminus W)$ is sum of edge variables with one endpoint in W and one in $V \setminus W$). This translates the constraints $y(W) \leq \frac{1}{2}(b(W) - 1)$ exactly to $y(W : V \setminus W) + s(W) \geq 1$. This means to find a violated constraint of this type it suffices to find W such that $y(W : V \setminus W) + s(W) < 1$.

Define a marked symmetric graph H over vertex set $U := V \cup \{z\}$ where z is a new vertex. Let H have symmetric capacity d : $d(u, v) := y_e$ when $u, v \in V$ and $u, v \in e$, and $d(u, v) := s_v$ when $u = z$ and $v \in V$. Let $M := \{v \in V \mid b_v \text{ is odd}\}$. If $|M|$ is odd, add z to M . Thus we have a marked symmetric graph $H = (U, d, M)$. Consider any odd marked cut W of H , without loss of generality $z \notin W$ (otherwise, take the complement). Observe that the value of edges crossing the cut is exactly $y(W : V \setminus W) + s(W)$; also note that $s(W)$ is odd. Thus there is a minimum odd marked cut W of H with value less than 1 iff there is a violated odd set constraint in (4).

By Theorem 22, there is a violated odd set constraint iff for some $s, t \in M$ the canonical minimum (s, t) -cut is an minimum odd marked cut with value less than 1. We conclude, using Theorem 17 and Lemma 18, that we can define a family of violated set constraints within FPC. Summing these defined violated constraints produces a canonical violated constraint which must be non-trivial by Proposition 6. Thus, as in Theorem 7 there is an FPC interpretation expressing the separation problem for the polytope in the linear program (4).

Lemma 24. *There is an FPC interpretation of τ_{vec} in $\tau_{match} \uplus \tau_{vec}$ expressing the separation problem for the b -matching polytopes with respect to their natural representation as τ_{match} -structures.*

Like the maximum flow problem in Section 5, the b -matching polytope is both compact and nonempty. By combining Lemma 24 and Theorem 9 with respect to the natural well-described representation of b -matching polytopes, we conclude that there is an FPC interpretation expressing the value of the maximum b -matching of a graph.

Theorem 25. *There is an FPC interpretation of $\tau_{\mathbb{Q}}$ in $\tau_{match} \uplus \tau_{vec}$ which takes a $\tau_{match} \uplus \tau_{vec}$ -structure coding a b -matching polytope P and a vector c to a rational number m indicating the value of the maximum b -matching of P with respect to c .*

9 Conclusion

Our main result is that the linear programming problem can be expressed in fixed-point logic with counting—indeed, that the linear optimisation problem can be expressed in FPC for any class of polytopes for which the separation problem can be defined in FPC. As a consequence, we solve an open problem of [BGS99] concluding that there is a formula of fixed-point logic with counting which defines the size of a maximum b -matching in a graph. This is one demonstration of the power of the ellipsoid method and linear optimisation that can be brought to bear even in the setting of logical definability. From here, there are number of natural research directions to consider.

Convex programming. A polytope is an instance of much more general geometric object: a convex set. The robust nature of the ellipsoid method means it has been extended to help solve more general optimisation problems, e.g., semi-definite programs and quadratic programs. It seems likely that our methods can be extended to these settings.

Completeness. Linear programming is complete for polynomial time under logspace reductions [DLR79]. It follows from our results that it cannot be complete for P under logical reductions such as first-order interpretations, since this would imply that P is contained in FPC. Could it still be the case that linear programming is complete for FPC under such weak reductions? Or perhaps FOC reductions? Even if linear programming is not complete, there may be other interesting combinatorial problems that can be expressed in FPC via reduction to linear programming. There has also been some work examining generalisations and improvements to the b -matching approach we followed (e.g., [CF96]), and it is possible these results can also be replicated in FPC.

LP hierarchies and integrality gaps. Another intriguing connection between counting logics and linear programming is established in [AM12, GO12] where it is shown that the hierarchy of Sherali-Adams relaxations [SA90] of the graph isomorphism integer program interleaves with equivalence in k -variable logic with counting (C^k). It is suggested [AM12] that inexpressibility results for C^k could be used to derive integrality gaps for such relaxations. It is a consequence of the results in this paper that the Sherali-Adams approximations of not only isomorphism, but of other combinatorial problems can be expressed in FPC. Do our results provide another route to using inexpressibility in FPC to prove integrality gaps?

Acknowledgments

The authors would like to thank Siddharth Barman for his helpful comments on an early draft of this paper and the anonymous reviewers for their constructive suggestions.

References

- [ABD09] A. Atserias, A. Bulatov, and A. Dawar, *Affine systems of equations and counting infinitary logic*, Theor. Comput. Sci. **410** (2009), no. 18, 1666–1683.
- [AM12] A. Atserias and E. Maneva, *Sherali-Adams relaxations and indistinguishability in counting logics*, ITCS, ACM, 2012, pp. 367–379.
- [BG05] A. Blass and Y. Gurevich, *A quick update on open problems in Blass-Gurevich-Shelah’s article ‘On polynomial time computations over unordered structures’*, Online at <http://research.microsoft.com/~gurevich/annotated.html>, 2005, [Accessed July 19, 2010].

- [BGS99] A. Blass, Y. Gurevich, and S. Shelah, *Choiceless polynomial time*, Ann. Pure Appl. Logic **100** (1999), 141–187.
- [BGS02] ———, *On polynomial time computation over unordered structures*, J. Symbolic Logic (2002), 1093–1125.
- [CF96] A. Caprara and M. Fischetti, $\{0, 1/2\}$ -Chvátal-Gomory cuts, Math. Program. **74** (1996), no. 3, 221–235.
- [CFI92] J-Y. Cai, M. Fürer, and N. Immerman, *An optimal lower bound on the number of variables for graph identification*, Combinatorica **12** (1992), no. 4, 389–410.
- [CH82] A. Chandra and D. Harel, *Structure and complexity of relational queries*, J. Comput. Syst. Sci. **25** (1982), no. 1, 99–128.
- [Dan63] G. Dantzig, *Linear programming and extensions*, Princeton University Press, 1963, (most recent edition published in 1998).
- [DGHL09] A. Dawar, M. Grohe, B. Holm, and B. Laubner, *Logics with rank operators*, LICS, IEEE, 2009, pp. 113–122.
- [DKL76] E.A. Dinitz, A.V. Karazanov, and M.V. Lomonosov, *On the structure of the system of minimum edge cuts in a graph*, Studies in Discrete Optimizations (1976), pp. 290–306 (Russian).
- [DLR79] D. Dobkin, R.J. Lipton, and S. Reiss, *Linear programming is log-space hard for P*, Inform. Process. Lett. **8** (1979), no. 2, 96–97.
- [Edm65] J. Edmonds, *Maximum matching and a polyhedron with 0, 1 vertices*, J. Res. Nat. Bur. Stand. **69 B** (1965), 125–130.
- [EF99] H.D. Ebbinghaus and J. Flum, *Finite model theory*, Springer, 1999.
- [GLS81] M. Grötschel, L. Lovász, and A. Schrijver, *The ellipsoid method and its consequences in combinatorial optimization*, Combinatorica **1** (1981), 169–197.
- [GLS88] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric algorithms and combinatorial optimization*, Springer-Verlag Berlin / New York, 1988.
- [GO12] M. Grohe and M. Otto, *Pebble Games and Linear Equations*, CSL, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012, pp. 289–304.
- [Gro10] M. Grohe, *Fixed-point definability and polynomial time on graph with excluded minors*, LICS, IEEE, 2010, pp. 179–188.
- [Hol10] B. Holm, *Descriptive complexity of linear algebra*, Ph.D. thesis, University of Cambridge, 2010.
- [Imm86] N. Immerman, *Relational queries computable in polynomial time*, Inform. Control **68** (1986), no. 1-3, 86–104.
- [Imm99] ———, *Descriptive complexity*, Springer-Verlag, 1999.

- [Kha79] L.G. Khachiyan, *A polynomial algorithm in linear programming.*, Dokl. Akad. Nauk SSSR **244** (1979), 1093–1096 (Russian).
- [Kha80] ———, *Polynomial algorithms in linear programming*, USSR Comp. Math. Math **20** (1980), no. 1, 53–72.
- [Lib04] L. Libkin, *Elements of finite model theory*, Springer, 2004.
- [PR82] M.W. Padberg and M.R. Rao, *Odd minimum cut-sets and b-matchings*, Math. Oper. Res. **7** (1982), no. 1, 67–80.
- [Ros10] B. Rossman, *Choiceless computation and symmetry*, Fields of Logic and Computation, Springer, 2010, pp. 565–580.
- [SA90] H.D. Sherali and W.P. Adams, *A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems*, SIAM Journal on Discrete Mathematics **3** (1990), no. 3, 411–430.
- [Sho72] N.Z. Shor, *Utilization of the operation of space dilatation in the minimization of convex functions*, Cybern. Syst. Anal. **6** (1972), no. 1, 7–15.
- [Sho77] ———, *Cut-off method with space extension in convex programming problems*, Cybern. Syst. Anal. **13** (1977), no. 1, 94–96.
- [ST04] D.A. Spielman and S.-H. Teng, *Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time*, JACM **51** (2004), no. 3, 385–463.
- [Var82] M. Vardi, *The complexity of relational query languages*, STOC, ACM, 1982, pp. 137–146.
- [YN76] D.B. Yudin and A.S. Nemirovskii, *Informational complexity and efficient methods for the solution of convex extremal problems*, Matekon **13** (1976), no. 2, 3–25.